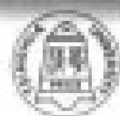
《最优化基础——模型与方法》系列教材

现代优化计算方法

邢文训 谢金星 编著



清华大学出版社
http://www.tup.tsinghua.edu.cn

《最优化基础——模型与方法》系列教材

现代优化计算方法

邢文训　谢金星　编著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书系统介绍了禁忌搜索、模拟退火、遗传算法、人工神经网络和拉格朗日松弛等现代优化计算方法的模型与理论、应用技术和应用案例。

全书共 6 章，第 1 章介绍算法复杂性的基本概念和启发式算法的评价方法，后 5 章分别介绍各个现代优化计算方法。

本书可作为数学、管理科学、计算机科学、工业工程等学科中相关优化专业的研究生教材，也可供相关专业研究人员参考。

图书在版编目(CIP)数据

现代优化计算方法/邢文训，谢金星编著. —北京：清华大学出版社，1999.8

"最优化基础——模型与方法"系列教材

ISBN 7-302-03610-1

Ⅰ. 现… Ⅱ. ① 邢…，② 谢… Ⅲ. 计算方法-最优化-高等学校-教材 Ⅳ. O241

中国版本图书馆 CIP 数据核字(1999)第 24726 号

出版者：清华大学出版社(北京清华大学校内，邮编 100084)

http://www.tup.tsinghua.edu.cn

印刷者：清华大学印刷厂

发行者：新华书店总店北京发行所

开　本：850×1168 1/32　印张：9.75　字数：252 千字

版　次：1999 年 8 月第 1 版　1999 年 8 月第 1 次印刷

书　号：ISBN 7-302-03610-1/O・215

印　数：0001～4000

定　价：13.50 元

《最优化基础——模型与方法》系列教材序言

最优化是人们在工程技术、科学研究和经济管理的诸多领域中经常遇到的问题。结构设计要在满足强度要求等条件下使所用材料的总重量最轻；资源分配要使各用户利用有限资源产生的总效益最大；安排运输方案要在满足物资需求和装载条件下使运输总费用最低；编制生产计划要按照产品工艺流程和顾客需求，尽量降低人力、设备、原材料等成本使总利润最高。可以预料，随着科学技术尤其是计算机技术的不断发展，以及数学理论与方法向各门学科和各个应用领域的更广泛、更深入的渗透，在即将到来的 21 世纪信息时代，最优化理论和技术必将在社会的诸多方面起着越来越大的作用。

解决实际生活中优化问题的手段大致有以下几种：一是靠经验的积累，凭主观作判断；二是做试验选方案，比优劣定决策；三是建立数学模型，求解最优策略。虽然由于建模时要作适当简化，可能使结果不一定非常完善，但是它基于客观数据，求解问题简便、灵活、经济，而且规模可以很大（将来会越来越大）。人们还可以吸收从经验得到的规则，用实验来不断校正建立的模型。随着数学方法和计算机技术的进步，用建模和数值模拟解决优化问题这一手段，将会越来越显示出它的效能和威力。显然，在决策定量化、科学化的呼声日益高涨的今天，数学建模方法的推广应用是符合时代潮流和形势发展需要的。

最优化理论、模型与方法所包含的内容很多，国内已出版了不少教材和专著介绍其各个分支。但是一方面，近年来发展起来的、有着广泛应用背景的规划模型（如随机规划、模糊规划等），以及一些已经为许多人采用、受到广泛关注的优化算法（如模拟退火、遗

传算法等），还缺乏详细和系统的介绍；另一方面，一些偏重优化理论和方法的教材，其要求难以与工科学生的数学知识衔接，也缺少对于应用来说十分重要的建模过程和软件介绍，而一些比较通俗的运筹学教材，则在加强理论基础，适应学生将来从事科研工作需要上考虑较少。我们这套教材试图弥补以上两方面的缺陷，力求体现下述特点：

1. 内容既包含传统的线性规划与非线性规划等部分，又纳入有广泛应用前景的随机规划和模糊规划；在传统内容中，既注重典型的数学思想和方法的系统叙述，又引入丰富的建模实例。

2. 数学基础既与工科学生所学知识衔接，又考虑到研究生阅读文献、从事科研工作的需要，适当提高理论基础的起点。

3. 对一般教材介绍的诸多算法进行精选，配合介绍一些应用软件，并引入近年来迅速发展的若干新算法。

本系列教材将陆续出版，首批四册为：《线性与非线性规划》、《网络优化》、《现代优化计算方法》、《随机规划与模糊规划》。

由于水平所限，书中难免有缺陷和错误，诚恳希望读者予以批评指正。

《最优化基础——模型与方法》系列教材编委会
1998年5月

序　言

随着20世纪80年代初期禁忌搜索、模拟退火、遗传算法和人工神经网络算法的兴起，科学工作者对这些算法的模型、理论和应用技术等一系列问题进行着深入的研究，并将这些算法称为现代优化算法。现代优化算法的主要应用对象是优化问题中的难解问题，也就是优化理论中的NP-hard问题。正是因为很多实际优化问题的难解性，和现代优化算法在一些优化问题中的成功应用，使得现代优化算法成为解决优化问题的一种有力工具。科学工作者对现代优化算法抱以极大的热情和期望，几乎在各种难解的优化问题中，他们都尝试这些算法的应用和比较其应用效果。

在国际和国内，每一种现代优化算法都有相应的专著，其中有较为详尽的理论和应用论述。很多科学工作者期望对这些算法有一定的了解，以在实际应用中有目的地采用。基于人们的这些期望和关注，我们将这些算法集于一书，从简单实例的应用，到其理论、应用技术及应用案例的深入分析，由易到难地向相关优化专业的硕士、博士研究生介绍现代优化算法的相关模型、理论、应用技术和一些应用实例，在国内，这可能是首次尝试。本书也可供科学研究人员在这些领域研究时参考。

我们不希望将现代优化算法中的诸多算法简单地堆集在一起，而是设法将它们有机地集于一书。基于这种想法，本书由6章组成。第1章介绍了现代优化算法要解决的问题及它们中的共同点，并将本书各章衔接在一起。第2章、第3章、第4章和第5章分

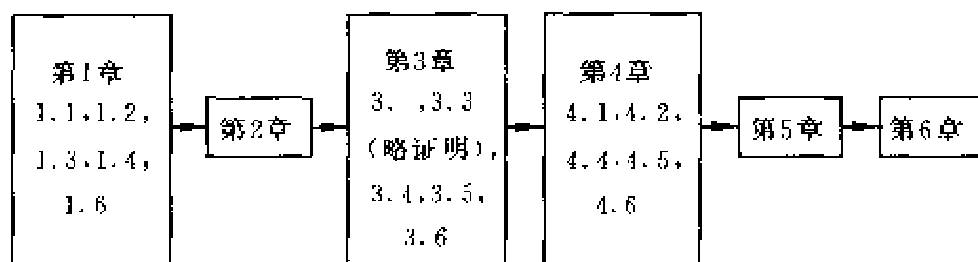
别介绍禁忌搜索算法、模拟退火算法、遗传算法和人工神经网络算法，这些是现代优化算法的组成。第6章提供评价算法的一种工具：拉格朗日松弛算法。

第1章为概论。首先介绍现代优化算法所要解决的组合优化问题。通过复杂性概念的引入，使得我们知道为什么和在什么情况下将现代优化算法应用到优化问题。通过邻域和算法评价方法的介绍，使我们找出现代优化算法的一些共同点。由于有关复杂性概念的内容不易理解，因此，作者在处理这部分内容时，以多个典型组合优化问题为背景，通过对它们的一步步分析来介绍复杂性的一个个概念。为了适应不同层次的读者，本章将复杂性概念的内容分为1.2节和1.5节两部分。1.2节介绍了多项式时间可求得最优解的多项式问题。1.5节更进一步介绍了NP、NP-complete和NP-hard概念。对学时要求较少或非运筹学专业学生的教学，可以略去1.5节。

将第2,3,4,5这四章的内容作为一个整体，从最容易理解的局部搜索算法开始，逐步深入地介绍全局搜索的禁忌搜索算法，带有随机搜索的模拟退火和遗传算法，最后，给出人工神经网络算法。对学时要求较少或非运筹学专业学生的教学，可以略去3.2节、3.3节和4.3节中的证明。

第6章拉格朗日松弛算法使得本书成为一个整体。我们不仅要学会应用现代优化算法，还应该学会评价这些算法。对于极小化目标函数的优化问题，现代优化算法能给出一个目标值不低于最优目标值的可行解，当评价一个算法的计算效率时，可行解目标值同最优目标值一个下界的差是评价的标准之一。拉格朗日松弛算法则是提供最优目标值下界的工具之一。

对学时要求较少或非运筹学专业学生的教学，可以参考下图教学内容：



本书是在1997年春季我校研究生“现代最优化算法”课程讲稿的基础上,经过1998年和1999年两次讲授后整理和扩充而成的。在此过程中,研究生们给予了非常有益的建议。自90年代初期,姜启源教授和韩继业教授投入了很大的心血,组织了组合优化问题讨论班并得到国家自然科学基金资助,这个讨论班和基金资助使我们积累了大量有价值的材料。在写作的过程中,香港城市大学的黎建强和林国健教授提供的资料大大丰富了书中的内容。谭泽光、胡冠章、林翠琴、刘宝碇教授和张家伟同学等阅读并修改了我们的手稿,给我们提出很多很好的建议。在此,对给予我们帮助的老师、出版社的编辑和同学们表示感谢!对给予我们部分资助的教育部高等学校工科数学教育基地表示感谢!最后,对我们家人的耐心和支持表示由衷的谢意!

由于我们的水平有限,恳请读者对本书的不足之处批评指正。

邢文训　谢金星

1999年春于清华园

目　　录

第1章
概　　论

现代优化算法包括禁忌搜索(tabu search)、模拟退火(simulated annealing)、遗传算法(genetic algorithms)、神经网络(neural networks)和拉格朗日松弛等算法.这些算法涉及生物进化、人工智能、数学和物理科学、神经系统和统计力学等概念,都是以一定的直观基础而构造的算法,我们称之为启发式算法.启发式算法的兴起与计算复杂性理论的形成有密切的联系.当人们不满足常规算法求解复杂问题时,现代优化算法开始体现其作用.现代优化算法自80年代初兴起,至今发展迅速.例如,在1996年,Osman在文[1]中分类罗列了1400篇相关文章.因此,这些算法同人工智能、计算机科学和运筹学相融合就不足为奇了.本章主要介绍组合优化问题、计算复杂性和启发式算法的概念.

1.1 组合最优化问题

组合最优化(combinatorial optimization)是通过对数学方法的研究去寻找离散事件的最优编排、分组、次序或筛选等,是运筹学(operations research)中的一个经典且重要的分支,所研究的问题涉及信息技术、经济管理、工业工程、交通运输、通信网络等诸多领域.该问题可用数学模型描述为:

$$\begin{aligned} &\min f(\boldsymbol{x}) \\ &\text{s.t.}\ g(\boldsymbol{x}) \geq 0, \\ &\quad \boldsymbol{x} \in D, \end{aligned}$$

其中，$f(x)$ 为目标函数，$g(x)$ 为约束函数，x 为决策变量，D 表示有限个点组成的集合.

一个组合最优化问题可用三参数(D,F,f)表示，其中 D 表示决策变量的定义域，F 表示可行解区域 $F=\{x|x\in D,\ g(x)\geqslant 0\}$，$F$ 中的任何一个元素称为该问题的可行解，f 表示目标函数. 满足 $f(x^*)=\min\{f(x)|x\in F\}$ 的可行解 x^* 称为该问题的最优解. 组合最优化的特点是可行解集合为有限点集. 由直观可知，只要将 D 中有限个点逐一判别是否满足 $g(x)$ 的约束和比较目标值的大小，该问题的最优解一定存在和可以得到. 因为现实生活中的大量优化问题是从有限个状态中选取最好的，所以大量的实际优化问题是组合最优化问题.

例 1.1 0-1 背包问题(knapsack problem)

设有一个容积为 b 的背包，n 个体积分别为 $a_i(i=1,2,\cdots,n)$，价值分别为 $c_i(i=1,2,\cdots,n)$ 的物品，如何以最大的价值装包？这个问题称为 0-1 背包问题. 用数学模型表示为：

$$\max \sum_{i=1}^{n} c_i x_i \tag{1.1}$$

$$\text{s.t.}\ \sum_{i=1}^{n} a_i x_i \leqslant b, \tag{1.2}$$

$$x_i \in \{0,1\},\ i=1,\cdots,n. \tag{1.3}$$

其中目标(1.1)欲使包内所装物品的价值最大，(1.2)为包的能力限制，(1.3)表示 x_i 为二进制变量，$x_i=1$ 表示装第 i 个物品，$x_i=0$ 表示不装. 此时，$D=\{0,1\}^n$，F 为 D 中满足(1.2)的可行解. f 为目标函数. □

例 1.2 旅行商问题(TSP，traveling salesman problem)

一个商人欲到 n 个城市推销商品，每两个城市 i 和 j 之间的距离为 d_{ij}，如何选择一条道路使得商人每个城市走一遍后回到起点且所走路径最短. TSP 还可以细分为对称和非对称距离两大类问

题. 当 $d_{ij}=d_{ji}, \forall\ i,j$ 时,称为对称距离 TSP,否则称非对称距离 TSP. 对一般的 TSP,它的一种数学模型描述为:

$$\min \sum_{i\neq j} d_{ij}x_{ij} \tag{1.4}$$

$$\text{s.t.} \sum_{j=1}^{n} x_{ij}=1,\ i=1,2,\cdots,n, \tag{1.5}$$

$$\sum_{i=1}^{n} x_{ij}=1,\ j=1,2,\cdots,n, \tag{1.6}$$

$$\sum_{i,j\in s} x_{ij}\leqslant |s|-1, 2\leqslant |s|\leqslant n-2, s\subset \{1,2,\cdots,n\}, \tag{1.7}$$

$$x_{ij}\in \{0,1\},\ i,j=1\cdots,n,\ i\neq j. \tag{1.8}$$

以上是基于图论的数学模型. 其中,(1.8) 中的决策变量 $x_{ij}=1$ 表示商人行走的路线包含从城市 i 到城市 j 路径,$x_{ij}=0$ 表示商人没有选择走这条路. $i\neq j$ 的约束可以减少变量的个数,使得共有 $n\times(n-1)$ 个决策变量. 目标(1.4)要求距离之和最小. (1.5) 要求商人从城市 i 出来一次,(1.6) 要求商人走入城市 j 只有一次. (1.5) 和(1.6) 表示每个城市经过一次. 仅有(1.5) 和(1.6) 的约束无法避免回路的产生,一条回路是由 $k(1\leqslant k\leqslant n)$ 个城市和 k 条弧组成,因此,(1.7) 约束旅行商在任何一个城市子集中不形成回路,其中 $|s|$ 表示集合 s 中元素个数. 此时,$D=\{0,1\}^{n\times(n-1)}$,F 为 D 中满足(1.5),(1.6) 和(1.7) 的可行解. f 为目标函数. □

上面两个问题都是组合优化中的经典问题. 它们的共性是读者容易接受这些问题,可行解集合是有限的,在问题的规模较小时,通过枚举很容易得到最优解.

例 1.3　整数线性规划(integer linear programming)

$$\text{(IP)}\qquad \begin{aligned} &\min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}, \end{aligned}$$

$$x \geqslant 0, x \in \mathbf{Z}^n.$$

其中,c 为 n 维列向量,A 为 $m \times n$ 矩阵,b 为 m 维列向量,x 为 n 维决策变量,$\mathbf{Z}^n$ 表示 n 维整数集合.模型中的系数 A、b 和 c 的元素都是整数. □

例 1.2 和 1.3 的数学模型都具有(IP)的形式.一些组合优化问题可以写成整数线性规划问题.整数线性规划同线性规划形式上非常相似,不同之处是前者的决策变量部分或全部取整数.

在本书中,我们假设线性整数规划的系数是整数.这主要是因为目前在计算机计算中都是以有限位符号储存和运算.很容易证明,IP 中的系数是有理数时都可以处理成整数情况.

例 1.4 装箱问题(bin packing)

如何以个数最少的尺寸为 1 的箱子装进 n 个尺寸不超过 1 的物品,该问题为装箱问题. □

例 1.5 约束机器排序问题(参考[2])(capacitated machine scheduling)

n 个加工量为 $\{d_i | i = 1,2,\cdots,n\}$ 的产品在一台机器上加工,机器在第 t 个时段的工作能力为 c_t,求完成所有产品加工的最少时段数.它的数学模型为

$$\min T \tag{1.9}$$

$$\text{s.t.} \sum_{t=1}^{T} x_{it} = 1,\ i = 1,2,\cdots,n, \tag{1.10}$$

$$\sum_{i=1}^{n} d_i x_{it} \leqslant c_t,\ t = 1,2,\cdots,T, \tag{1.11}$$

$$x_{it} \in \{0,1\},\ i = 1,2,\cdots,n;\ t = 1,2,\cdots,T, \tag{1.12}$$

其中,x_{it},T 为决策变量,$x_{it} = 1$ 表示第 t 时段加工产品 i.(1.9) 要求加工所用的时段数最少,(1.10) 表示产品 i 一定在某一个时段加工,(1.11) 表示每个时段的加工量不超过能力的限制. □

有些优化问题可以用整数规划模型的形式表示,如例 1.1 和

1.2.有些组合优化问题用整数规划模型表示则比较复杂和不易被理解,不如对问题采用直接叙述更易理解,如例1.2、1.4和1.5.因此,有大量的组合优化问题是通过文字语言叙述的.

1.2 计算复杂性的概念

由组合最优化问题的定义可知,每一个组合最优化问题都可以通过枚举的方法求得最优解.枚举是以时间为代价的,有的枚举时间还可以接受,有的则不可能接受.如例1.2的非对称距离TSP问题,可以用另一个方法来表示它的可行解:用n个城市的一个排列表示商人按这个排列序推销并返回起点.若固定一个城市为起终点,则需要$(n-1)!$个枚举.以计算机1秒可以完成24个城市所有路径枚举为单位,则25个城市的计算时间为:以第1个城市为起点,第2个到达城市有可能是第2、第3、…、或第25个城市.决定前两个城市的顺序后,余下是23个城市的所有排列,枚举这23个城市的排列需要1秒,所以,25个城市的枚举需要24秒.类似地归纳,城市数同计算时间的关系如表1.1所示.

表1.1 枚举时城市数与计算时间的关系

城市数	24	25	26	27	28	29	30	31
计算时间	1s	24s	10min	4.3h	约4.9d	136.5d	约10.8a	约325a

通过表1.1可以看出,随着城市数的增多,计算时间增加非常之快,当城市数增加到30时,计算时间约10.8年,已无法接受.

问题是一个抽象的模型或概念,它通过一些具体的数据表现出来.问题是需要回答的一般性提问,通常含有若干个满足一定条件的参数.问题通过下面的描述给定:(1)描述所有参数的特性,(2)描述答案所满足的条件.如TSP是一个问题,它由例1.2的文

字或模型(1.4)～(1.8)表示，通过给出的城市数和城市间的距离而使这个问题具体化．目前计算机可以求解的是这些给出城市数和城市间的距离的具体 TSP．一个算法常常是针对一个问题来设计的，如上面的 TSP 枚举算法可以求解任何一个 TSP 的最优解．而若用计算机求解，则必须对问题中的参数赋予具体值，如 TSP 中的城市数和城市间的距离，问题中的参数赋予了具体值的例子称为问题的实例(instance)．一个问题通过它的所有实例表现．

衡量一个算法的好坏通常是用算法中的加、减、乘、除和比较等基本运算的总次数同实例的计算机计算时的二进制输入数据的大小关系来度量．对于一个正整数 x，二进制表示是以

$$x = a_s 2^s + a_{s-1} 2^{s-1} + \cdots + a_1 \times 2 + a_0 \quad (a_s \neq 0)$$

的系数$(a_s a_{s-1} \cdots a_1 a_0)$来表示．由

$$2^s \leqslant 2^{\log_2 x} = x \leqslant 2^{s+1} \leqslant 2^{\log_2 x+1},$$

这个二进制数的位数是 $s+1$，也称为数据 x 的输入长度，在 $\log_2 x$ 与 $\log_2 x + 1$ 之间．于是，x 的二进制数的位数是整数$\lceil \log_2 x \rceil$或$\lceil \log_2 x \rceil + 1$，但不超过$\lceil \log_2 x \rceil + 1$，其中$\lceil x \rceil$表示不小于 x 的最小整数．虽说 $\log_2 0$ 没有意义和 $\log_2 1 = 0$，特别需要注意的是整数 0 和 1 的二进制位数都是 1．可以看到，任何一个非负整数的输入长度至少为 1．在此，在特别定义 $\log_2 0 = 0$ 后，上面的二进制表示方法和输入长度的结论可以推广到非负整数．常规理解的计算机只能用有限位表示一个实数，因此，我们限定只考虑有理数．

如表 1.1 对应的 TSP，当城市数为 n 且第一个城市为始终点时，计算一条路径$(1, i_2, \cdots, i_n)$长度的基本运算为两两城市间距离的 n 个求和．因为有$(n-1)!$条路径，枚举所有路径进行$(n-1)!$次比较，可以得到最优路径．这个枚举算法的基本计算总次数为$\{(n-1)!\}n = n!$．实例的计算机计算时只需输入城市数和任何两个城市的距离．于是，输入数据是城市数 n 和城市间的距离 $S = \{d_{ij} \mid 1 \leqslant i, j \leqslant n, i \neq j\}$，且假设这些数为整数．对任何一个非零距

离 d_{ij}，它的二进制输入数据的输入长度不超过 $\log_2|d_{ij}|+1$，可以用一个稍大的整数估计为$\lceil \log_2|d_{ij}| \rceil+1$，数字 0 可以用一位二进制码表示. 得到实例的二进制输入长度总量不超过

$$L = n(n-1) + \log_2|P|,$$

其中，$P = \Pi\{d_{ij}|d_{ij} \neq 0\}$.

若假设 $S = \{d_{ij}|1 \leqslant i,j \leqslant n, i \neq j\}$ 中每一个数据的绝对值都有上界 K，则有 $L = n(n-1) + \log_2|P| \leqslant n(n-1)(1+\log_2 K)$. 此时发现，TSP 实例的计算机二进制输入长度和算法的基本计算总次数都是城市数 n 的函数. TSP 实例的二进制输入长度是 n 的多项式函数. 枚举算法的基本计算总是 n 的阶乘函数，比指数函数增加的速度还快.

我们对实例的二进制输入长度和算法的基本计算总次数是粗略估计的，一般是给予一个上限. 一个求解实例 I 的算法的基本计算总次数 $C(I)$ 同实例 I 的计算机二进制输入长度 $d(I)$ 的关系常用符号 $C(I) = f(d(I)) = O(g(d(I)))$ 表示，它的含义是：求解实例 I 的算法的基本计算总次数 $C(I)$ 是实例输入长度 $d(I)$ 的一个函数，这个函数被另一个函数 $g(x)$ 控制，即存在一个函数 $g(x)$ 和一个常数 α，使得

$$C(I) \leqslant \alpha g(d(I)). \tag{1.13}$$

(1.13) 表示算法在求解实例 I 时，所用的基本计算总次数可以被一个函数值 $g(d(x))$ 控制. $g(x)$ 的函数特性决定了基本计算总次数的性能.

定义 1.1　假设问题和解决该问题的一个算法已经给定，若给定该问题的一个实例 I，存在多项式函数 $g(x)$，使得(1.13) 成立，我们称该算法对实例 I 是多项式时间算法；若存在 $g(x)$ 为多项式函数且对该问题任意的一个实例 I，都有(1.13) 成立，则称该算法为解决该问题的多项式时间算法. 特别对优化问题，当一个算法为求解该优化问题最优解的多项式时间算法，则称为该优化问

题的多项式时间最优算法.

为了方便,当 $g(x)$ 为指数函数时,称相应的算法是指数时间算法.相比较而言,随着变量的增加,多项式函数增长的速度比指数函数增长的速度要慢得多,因此,我们更喜欢多项式函数.例如,随着 n 的增加,n^k($k > 0$ 的整数)的增长速度远比 a^n($a > 1$ 为实数)增长的速度慢.若一个多项式时间算法的基本计算总次数为 $L_1 = 2^{10}n^2$,而另一个非多项式时间算法的计算次数为 $L_2 = 2^n$.当 n 比较小的时候,多项式时间算法的优越性并不明显.如当 $n = 10$ 时,$L_1/L_2 \approx 10^2$,多项式时间算法的计算次数是非多项式时间算法的 10^2 倍.但当 $n = 2^5$ 时,$L_1/L_2 = 2^{-12}$,非多项式时间算法的计算次数上升非常快,是多项式时间算法的计算次数的 2^{12} 倍.

由上面的讨论看出,在实例的二进制输入规模增大时,多项式时间算法的基本计算总次数的增加速度相对较慢.目前习惯将算法分为多项式时间算法和非多项式时间算法.这种归类方法主要考虑的是 n 增加时的变化趋势.重述一个观点,我们构造算法的目的是能够解决问题的所有实例而不单单是问题的一个实例.根据这个观点,再结合问题、实例和算法,形成了复杂性分析的概念.

复杂性分析的一个研究方向是对算法进行评价.对于解决一个问题的算法,如何评估这个算法的性能?复杂性分析是评价算法的一个指标.复杂性分析是通过(1.13)从最坏实例的条件下,确定是否存在多项式函数 $g(x)$,即可以叙述为:对一个求解问题的算法,是否存在多项式函数 $g(x)$ 和常数 α,使得对问题的任意一个实例 I,都有(1.13)成立.

以线性规划中的单纯形算法为例.先估计线性规划任何一个实例的二进制输入长度.

例 1.6 线性规划问题(linear programming)

$$(\mathrm{LP}) \qquad \begin{cases} \quad \min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ \quad \boldsymbol{x} \geqslant 0, \boldsymbol{x} \in \mathbf{R}^n. \end{cases}$$

其中，$\boldsymbol{c}$ 为 n 维列向量，$\boldsymbol{A}$ 为 $m\times n$ 矩阵，$\boldsymbol{b}$ 为 m 维列向量，$\boldsymbol{A},\boldsymbol{b},\boldsymbol{c}$ 各分量为整数，$\boldsymbol{x}$ 为 n 维决策变量. 由于任何一个非负整数 x 的输入长度至少为1且不超过 $\log_2 x+1$，(LP) 的三组数据 $\boldsymbol{c}=(c_1,c_2,\cdots,c_n)^{\mathrm{T}}$，$\boldsymbol{A}=(a_{ij})_{mn}$，$\boldsymbol{b}=(b_1,b_2,\cdots,b_m)^{\mathrm{T}}$ 的二进制输入长度的下上限分别是下列区间的左右端点

$$\left[\max\left\{n,\sum_{i=1}^{n}\log_2|c_i|\right\},\sum_{i=1}^{n}(\log_2|c_i|+1)\right],$$

$$\left[\max\left\{mn,\sum_{i=1}^{m}\sum_{j=1}^{n}\log_2|a_{ij}|\right\},\sum_{i=1}^{m}\sum_{j=1}^{n}(\log_2|a_{ij}|+1)\right]$$

和

$$\left[\max\left\{m,\sum_{i=1}^{m}\log_2|b_i|\right\},\sum_{i=1}^{m}(\log_2|b_i|+1)\right].$$

这三组数据输入长度累计之和的上下限分别为 $mn+m+n+\log_2|p|$ 和 $\max\{mn+m+n,\log_2|p|\}$，其中 p 为 $\boldsymbol{A}$、$\boldsymbol{b}$ 和 $\boldsymbol{c}$ 中所有非零数的乘积. □

单纯形算法是解决线性规划问题的主要算法之一，对于大多数实例，它的计算效果非常好，这可从实际工程、管理人员的大量实际应用得以证实. 也可以说，线性规划问题的很多实例存在多项式时间的算法，但也存在极端的情形，Klee 和 Minty 在文献[3]中给出下面这样一个实例

$$\begin{cases}\min -x_n\\ \text{s.t. } x_1-r_1=\dfrac{1}{4},\\ \qquad x_1+s_1=1,\\ \qquad x_j-\dfrac{1}{4}x_{j-1}-r_j=0,\quad j=2,3,\cdots,n,\\ \qquad x_j+\dfrac{1}{4}x_{j-1}+s_j=1,\quad j=2,3,\cdots,n,\\ \qquad x_j,r_j,s_j\geqslant 0,\quad j=1,2,\cdots,n.\end{cases}\tag{1.14}$$

文献[3]中证明用单纯算法求解线性规划(1.14)，其迭代步数是 2^n-1. 由于实例(1.14)共有 $2n$ 个约束和 $3n$ 个变量，从例1.6对实例二进制输入大小的估计得到，二进制输入大小 $d(I)$ 不超过 $6n^2+2n+3n+\log_2|p|$，其中 p 是(1.14)中所有非零系数的乘积.

由此回答，对线性规划的单纯形算法，不存在多项式函数 $g(x)$ 和常数 α，使得对问题的任意一个实例 I，都有(1.13)成立. 本章的第4节将进一步讨论有关算法的评价方法.

复杂性分析的另一个研究方向是对组合优化问题归类. 可以这样定义多项式问题.

定义 1.2 对于给定的一个优化问题，若存在一个求解该问题最优解的算法、一个多项式函数 $g(x)$ 和常数 α，使得(1.13)对给定优化问题的任何一个实例成立，则称给定的优化问题是多项式时间可解问题，或简称多项式问题，所有多项式问题集记为P(polynomial).

还以例1.6的线性规划问题为例，上面已经得到单纯形算法不是线性规划问题的多项式时间算法，但这并不能说明线性规划问题不属于多项式问题. Khachian 在文献[4]中成功地构造了椭球算法并证明了其算法是线性规划问题的多项式时间算法. 因此，线性规划问题是多项式问题.

并不是所有的组合优化问题都找到了求解最优解的多项式时间算法. 也就是说，还不能肯定一些组合优化问题是否属于P. 经过几代数学家的努力，他们研究、整理了一类难以求解的组合最优化问题，迄今为止，这些问题还没有一个有能求得最优解的多项式时间算法. 这一类组合最优化问题归为所谓的 NP-hard. 受人类认识能力的限制，目前人们只能假设这一类难解的组合最优化问题不存在求解最优解的多项式时间算法. 本章第1节中例1.1、例1.2、例1.3、例1.4和例1.5的组合最优化问题都属于这类难解的

问题.有关复杂性的一些定义和一个问题复杂性的确定将在本章的第 5 节进一步讨论.感兴趣的读者可以继续阅读这一节或参考文献[5].

正因为一些组合最优化问题还没有找到求最优解的多项式时间算法,而这些组合最优化问题又有非常强的实际应用背景,人们才不得不尝试用一些并不一定可以求解到最优解的算法,在此称为启发式算法,来求解组合最优化问题.

1.3　邻域概念

在距离空间中,通常的邻域定义是以一点为中心的一个圆.光滑函数极值的数值求解中,邻域是一个非常重要的概念,函数的下降或上升都是在一点的邻域中寻求变化方向.在组合优化中,距离的概念通常不再适用,但是在一点附近搜索另一个下降的点仍然是组合优化数值求解的基本思想.因此,需要重新定义邻域的概念.

定义 1.3　对于组合优化问题(D,F,f),D 上的一个映射:

$$N:\ S \in D \to N(S) \in 2^D$$

称为一个邻域映射,其中 2^D 表示 D 的所有子集组成的集合,$N(S)$ 称为 S 的邻域,$S' \in N(S)$ 称为 S 的一个邻居.

例 1.7　例 1.2 已给出 TSP 的一种数学模型,由模型 $D = \{x \mid x \in \{0,1\}^{n\times(n-1)}\}$,可以定义它的一种邻域为:

$$N(x) = \left\{y \,\middle|\, |y - x| = \sum_{i,j} |y_{ij} - x_{ij}| \leqslant k, y \in D\right\},$$

k 为一个正整数.

这个邻域定义使得 x 最多有 k 个位置的值可以发生变化,x 的邻居有 $C_{n(n-1)}^1 + C_{n(n-1)}^2 + \cdots + C_{n(n-1)}^k$ 个.　□

例 1.8　TSP 问题解的另一种表示法为 $D = F = \{S = (i_1, i_2,$

$\cdots,i_n)|i_1,i_2,\cdots,i_n$ 是 $1,2,\cdots,n$ 的一个排列}. 文献[6]中定义它的邻域映射为 2-opt，即 S 中的两个元素进行对换，$N(S)$ 中共包含 S 的 C_n^2 个邻居. 如四个城市的 TSP 问题，当 $S=(1,2,3,4)$ 时，$N(S)=\{(2,1,3,4),(3,2,1,4),(4,2,3,1),(1,3,2,4),(1,4,3,2),(1,2,4,3)\}$. □

类似 2-opt 的定义，可以推广定义 k-opt($k\geqslant 2$)，它的邻域映射是对 S 中的 k 个元素按一定的规则互换.

例 1.9 0-1 背包问题. 该问题解的另一种表示法为 $D=\{(i_1,i_2,\cdots,i_n)|i_1,i_2,\cdots,i_n$ 是 $1,2,\cdots,n$ 的一个排列$\}$. $(i_1,i_2,\cdots,i_n)$ 表示装包的排列顺序. 通过排列顺序以容量约束判别装进包的物品及目标值. 由该法定义的邻域可以同上例有相同的结构. □

邻域的构造依赖于问题决策变量的表示，邻域的结构在现代优化算法中起很重要的作用，这一点将在以后的各章节中看出. 有邻域的定义后，类似连续函数，可以定义局部、全局最优概念.

定义 1.4 若 s^* 满足

$$f(s^*)\leqslant(\geqslant)f(s), \text{其中}, s\in N(s^*)\cap F,$$

则称 s^* 为 f 在 F 上的局部(local)最小(最大)解. 若

$$f(s^*)\leqslant(\geqslant)f(s), s\in F,$$

则称 s^* 为 f 在 F 上的全局(global)最小(最大)解.

就一维变量 x 为例，定义域为区间[1,10]中的整数点，像图 1.1 一样，如果采用如下邻域定义

$$N(x)=\{y\in Z_+ \mid |y-x|\leqslant 1\},$$

目标值如图 1.1，则 $x=9$ 为 f 的全局最优(最小)点，$x=5$ 为 f 的局部最优(最小)点，而 $x=4$ 点即不是 f 的局部最大也不是局部最小.

在求解最小目标值点时，传统的优化算法是以一个初始点出发，在邻域中寻找目标值更小的点，最后达到一个无法再下降的点. 如图 1.1，若以 $x=4$ 为起点按传统的优化方法搜索最小值点，

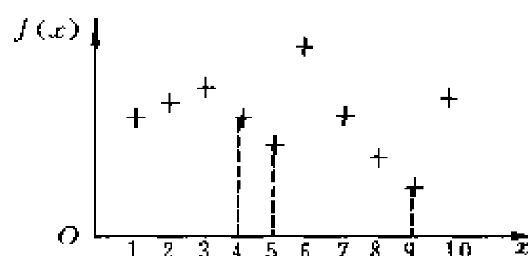


图 1.1　局部最优示意图

则搜索到 $x=5$ 而停止，搜索到局部最优(最小)点. 这种方法可能造成最终解的非全局最优性. 现代优化算法所要解决的一个问题就是求解全局最优解.

1.4　启发式算法

启发式算法(heuristic algorithm) 是相对于最优算法提出的. 一个问题的最优算法求得该问题每个实例的最优解. 启发式算法可以这样定义:

一个基于直观或经验构造的算法，在可接受的花费(指计算时间、占用空间等) 下给出待解决组合优化问题每一个实例的一个可行解，该可行解与最优解的偏离程度不一定事先可以预计.

另一种定义为:启发式算法是一种技术. 这种技术使得在可接受的计算费用内去寻找最好的解，但不一定能保证所得解的可行性和最优性，甚至在多数情况下，无法阐述所得解同最优解的近似程度(参考文献[7]).

在某些情况下，特别是实际问题中，最优算法的计算时间使人无法忍受或因问题的难度使其计算时间随问题规模的增加以指数速度增加，如表 1.1 列举的 TSP 变化情况，此时只能通过启发式算法求得问题的一个可行解. 启发式方法是对启发式算法的综合

应用.

上述启发式算法的两个定义中,一个共同的特点是不考虑算法所得解与最优解的偏离程度.如果采用分析最坏情况的误差界限的概念来评价启发式算法,则可以将近似算法定义如下.

定义 1.5 设 A 是一个问题.记问题 A 的任何一个实例 I 的最优解和启发式算法 H 解的目标值分别为 $z_{OPT}(I)$ 和 $z_H(I)$.于是对于某个 $\epsilon \geqslant 0$,称 H 是 A 的 ϵ 近似算法(ϵ-approximation algorithm)当且仅当

$$|z_H(I) - z_{OPT}(I)| \leqslant \varepsilon |z_{OPT}(I)|, \quad \forall I \in A.$$

上面的定义给出启发式算法和近似算法两个概念的联系和区别.启发式算法概念定义的算法集合包含了近似算法概念定义的算法集合.近似算法强调给出算法最坏情况的误差界限,而启发式算法不需考虑偏差程度.由于很多组合优化问题算法的最坏情况误差界估计需要很强的数学基础和较强的技巧,甚至一些问题很难或无法给出最坏情况误差界,而实际问题又迫切需要求解的方法,因此,只能通过启发式算法解决问题.

早在 40 年代末期,由于实际问题的需要,人们已提出一些解决实际问题的快捷有效的启发式算法,有代表性的工作是 1948 年 Polya 的著作[8].随着 60—70 年代对数学模型及最优解算法的重视,这些算法被称为"快速与丑陋(quick and dirty)"方法.随着 70 年代算法复杂性理论的完善,我们不再强调一定要求到最优解,因此促使 80 年代初兴起的现代优化算法在今天得到了巨大的发展.

让我们通过下面的例子对启发式算法有一定的了解.

例 1.10 背包问题的贪婪算法(greedy algorithm)

对例 1.1 的背包问题可以构造下面的贪婪算法

STEP1 对物品以 $\frac{c_i}{a_i}$ 从大到小排列,不妨把排列记成{1,2,…,

$n\}$, $k := 1$;

STEP2　若 $\sum_{i=1}^{k-1} d_i x_i + d_k \leqslant b$,,则 $x_k = 1$; 否则,$x_k = 0$. $k := k + 1$;

当 $k = n + 1$ 时,停止;否则,重复 STEP2.

$(x_1, x_2, \cdots, x_n)$ 为贪婪算法所得解. 单位体积价值比越大越先装包是贪婪算法的原则. □

这样的算法非常直观,非常容易操作.

例 1.11　简单的邻域搜索(local search) 算法

给定组合优化问题,假设其邻域结构已确定. 设 S 为解集合,f 为 S 上的费用函数,N 为邻域结构. 算法为:

STEP1　任选一个初始解 $s_0 \in S$;

STEP2　在 $N(s_0)$ 中按某一规则选一 s;若 $f(s) < f(s_0)$,则 $s_0 := s$;否则,$N(s_0) := N(s_0) - s$;若 $N(s_0) = \varnothing$,停止;否则,返回 STEP2.

简单的邻域搜索从任何一点出发,达到一个局部最优值点. 从算法中可以看出,算法停止时得到点的性质依赖算法初始解的选取、邻域的结构和邻域选点的规则. 一个直观的看法是:只要选好初始点,就一定可以求到最优解. 对 NP-hard 的组合最优化问题,确定这样的初始点是非常困难的,如非对称 TSP 问题,若固定一个城市为起终点,共有 $(n-1)!$ 个解,也就是有 $(n-1)!$ 个初始点. 如何选初始点和如何跳出局部最优值点以达最优点是本书后续部分的一个主要内容. □

启发式算法能够迅速发展是因为它有以下长处:

(1) 数学模型本身是实际问题的简化,或多或少地忽略一些

因素;数据采集具有不精确性;参数估计具有不准确性;以上因素可能使最优算法所得解比启发式算法所得解产生更大误差.

(2) 有些难的组合优化问题可能还没有找到最优算法,即使存在,由算法复杂性理论,得知它们的计算时间是无法接受或不实际的.

(3) 一些启发式算法可以用在最优算法中,如在分支定界算法中,可以用启发式算法估界.

(4) 简单易行;比较直观;易被使用者接受.

(5) 速度快,在适时管理中非常重要.

(6) 多数情况下,程序简单,因此易于修改.

虽说有诸多好处,但它有其短处,这些不足往往成为争论的焦点.

(1) 不能保证求得最优解.

(2) 表现不稳定.启发式算法在同一问题的不同实例计算中会有不同的效果.有些解很好,而有些则很差.在实际应用中,这种不稳定性造成计算结果不可信.

(3) 算法的好坏依赖于实际问题、经验和设计者的技术,这一点很难总结规律,同时使不同算法之间难以比较.

1.4.1 启发式算法的分类

本节对启发式算法进行简单的分类.

1. 一步算法

该算法的特点是:不在两个可行解之间选择,在未终止的迭代中,有可能不是一个可行解,算法结束时得到一个可行解.

一步算法的一个典型实例是背包问题的贪婪算法.每一步迭代选一物品入包,直到无法再装.该算法没有在两个可行解之间比较选择,算法结束时得到一可行解.

再以 TSP 中的路线搜索算法来解释上面的归类.

例 1.12 TSP 的路线搜索算法为：

STEP1	$P = \{1\}$; $N = \{1,2,\cdots,n\}$; $i = 1$;
STEP2	从$\{d_{ij} \mid i$ 可达 j，但 $j \notin P\}$ 选最小的值 d_{ii_0}，对应城市为 i_0，置 $P = P \cup \{i_0\}$，若 $P = \{1,2,\cdots,n\}$，停止，否则 $i = i_0$，重复 STEP2.

该算法的特征是每一步选择一个城市，算法没结束前没有得到可行解. 算法结束时，可行解得到. 同样，没有在两个可行解之间选择. □

2. 改进算法

改进算法的迭代过程是从一个可行解到另一个可行解，通常通过两个解的比较而选择好的解，进而作为新的起点进行新的迭代，直到满足一定的要求为止. 因此，也可以称之为迭代算法. 如例 1.11 的局部搜索算法. 再如 TSP 中简单的 2-opt 方法[6]，也是改进算法的一种.

例 1.13 四个城市$\{1,2,3,4\}$的TSP问题，两个城市 i 和 j 之间的距离 d_{ij} 构成的距离矩阵为

$$(d_{ij}) = \begin{bmatrix} 0 & 5 & 10 & 15 \\ 5 & 0 & 7 & 8 \\ 10 & 7 & 0 & 6 \\ 15 & 8 & 6 & 0 \end{bmatrix}$$

设初始点为：$s_0 = (1,2,3,4)$，则

$$f(s_0) = d_{12} + d_{23} + d_{34} + d_{41} = 5 + 7 + 6 + 15 = 33.$$

s_0 的 2-opt 邻域是对 s_0 的任两个元素对换，如果固定第一个城市 1，即 $N(s_0) = \{(1,3,2,4), (1,4,3,2), (1,2,4,3)\}$，简单的 2-opt 方法是比较邻域中的所有点，选出最好解. 比较可知 $N(s_0)$ 中最好的解为 $s_1 = (1,2,4,3)$，目标值为 $f(s_1) = 29$. 下一次迭代是以 s_1

为起点，重复以上的计算过程，直到目标值无法改进为止. □

3. 数学规划算法

数学规划算法主要指用线性规划的方法求解组合优化问题，其中包括一些启发式规则. 这一类方法中，典型的是线性规划及对偶理论在网络流中的应用，产生标号算法等一系列最优算法，参考文献[9,10]. 其次是基于整数规划分支定界的启发式算法，只搜索一些特殊分支，或是基于整数规划的割平面法，产生的考虑部分割平面的算法.

4. 解空间松弛算法

一类方法是线性规划松弛. 这类方法的主要步骤是先将整数规划问题 IP

$$(\mathrm{IP})\qquad \begin{cases} z = \min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \\ \qquad \boldsymbol{x} \geqslant \boldsymbol{0}, \boldsymbol{x} \in \boldsymbol{Z}^n \end{cases}$$

松弛为线性规划 LP

$$(\mathrm{LP})\qquad \begin{cases} z_L = \min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \\ \qquad \boldsymbol{x} \geqslant \boldsymbol{0}. \end{cases}$$

通过LP可以得到IP的一个下界 $z_L \leqslant z$. LP是多项式可解问题，但 LP 的解不一定是 IP 的可行解[4]. 于是，解空间松弛算法的第二步是如何将 LP 的解转化为 IP 的可行解，如四舍五入法，取上整数、下整数等.

拉格朗日(Lagrange)松弛法. 拉格朗日松弛法主要用于求解下面这样的组合最优化问题 IP

$$z = \min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \tag{1.15}$$

$$\text{s.t. } \boldsymbol{A}_1\boldsymbol{x} \geqslant \boldsymbol{b}_1, \tag{1.16}$$

$$\boldsymbol{A}_2\boldsymbol{x} \geqslant \boldsymbol{b}_2, \tag{1.17}$$

$$\boldsymbol{x} \geqslant \boldsymbol{0}, \boldsymbol{x} \in \boldsymbol{Z}^n, \tag{1.18}$$

如果没有(1.17)这些约束,IP问题是一个多项式时间问题,有(1.17)这些约束,IP为NP-hard问题.通常称(1.17)为难约束.这类方法的主要步骤分三个阶段.第一阶段是先将整数规划问题IP松弛为拉格朗日松弛问题LR

$$z_{LR}(\boldsymbol{\lambda}) = \min \boldsymbol{c}^{T}\boldsymbol{x} + \boldsymbol{\lambda}^{T}(\boldsymbol{b}_2 - \boldsymbol{A}_2\boldsymbol{x}) \tag{1.19}$$

$$\text{s.t. } \boldsymbol{A}_1\boldsymbol{x} \geqslant \boldsymbol{b}_1, \tag{1.20}$$

$$\boldsymbol{x} \geqslant \boldsymbol{0}, \boldsymbol{\lambda} \geqslant \boldsymbol{0}, \boldsymbol{x} \in \boldsymbol{Z}^n. \tag{1.21}$$

(1.17)的存在使得问题IP成为NP-hard问题.松弛(1.17)约束后,IP成为一个易解的问题.(1.17)松弛后,IP的解空间扩大,于是将(1.17)以罚函数的形式在LR的(1.19)体现,不至于得到的解使(1.17)的不可行性过大.在这一阶段中,提供了IP的一个下界$z_{LR}(\boldsymbol{\lambda})$.

第二阶段是求IP对偶问题的解,即

$$z_{LD} = \max_{\boldsymbol{\lambda} \geqslant 0} z_{LR}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda} \geqslant 0} \min_{\{\boldsymbol{x} \mid \boldsymbol{A}_1\boldsymbol{x} \geqslant \boldsymbol{b}, \boldsymbol{x} \geqslant \boldsymbol{0}, \boldsymbol{x} \in \boldsymbol{Z}^n\}} \boldsymbol{c}^{T}\boldsymbol{x} + \boldsymbol{\lambda}^{T}(\boldsymbol{b}_2 - \boldsymbol{A}_2\boldsymbol{x}). \tag{1.22}$$

通过第二阶段的求解,得到(1.22)的解$\boldsymbol{x}$和$\boldsymbol{\lambda}$,且有$z \geqslant z_{LD} \geqslant z_{LR}(\boldsymbol{\lambda})$.从理论上可以讨论$z$同$z_{LD}$、$z$同$z_{LR}(\boldsymbol{\lambda})$、$z_{LR}(\boldsymbol{\lambda})$同$z_{LD}$等的差距,这些内容将在第6章——拉格朗日松弛算法中讨论.

第三个阶段将第二阶段所得解的可行化,使之成为IP的可行解.可行化的方法往往依赖于问题本身,这些内容也将在第6章讨论.

5. 现代优化算法

现代优化算法是80年代初兴起的启发式算法.这些算法包括禁忌搜索(tabu search),模拟退火(simulated annealing),遗传算法(genetic algorithms),人工神经网络(neural networks).它们主要用于解决大量的实际应用问题.目前,这些算法在理论和实际应用方面都得到了较大的发展.无论这些算法是怎样产生的,它们有

一个共同的目标 —— 求NP-hard组合优化问题的全局最优解.虽说有这样的目标,但NP-hard理论限制它们只能以启发式的算法去求解实际问题.本书将以这几类算法为重点,从基本理论、应用技术和应用实例等方面讲解.

6. 其他方法

启发式方法包含的类型很多,有些方法是根据实际问题而产生.如解空间分解、解空间的限制等.另一类算法是集成算法,这些算法是诸多启发式算法的合成.

1.4.2 启发式算法的性能分析

虽说启发式算法有诸多优点,但它的一个缺陷是无法保证得到最优解,于是,对算法的评价显得尤为重要.一个好的启发式算法可以使其解同最优解尽可能地接近,同时保证有较好的稳定性.评价启发式算法的性能有不同的方法,主要分为对算法计算复杂性的分析和对算法计算效能的分析.下面仅简单介绍常用的方法.

1. 最坏情形分析

最坏情形分析(worst case analysis)可以考虑计算复杂性和计算解的效果两个方面.最坏情形计算复杂性分析关注算法基本运算总次数同实例计算机二进制输入长度之间的关系,从最坏实例的角度来研究算法的计算时间复杂性.这一部分内容已在第2节讨论,在这里不再讨论.通过下面的一个例子了解对一些比较简单的启发式算法是如何估计计算时间的复杂性.

例 1.14 估计例1.10的贪婪算法的计算时间复杂性.

它的STEP1需要n个除法和将n个比值$\{\frac{c_i}{a_i}|i=1,2,\cdots,n\}$从大到小排序.STEP2中的每一个循环最多需要$k-1$个乘法、$k$个加法、1个比较和1个赋值($x_k=1$,等价一个加法),全部循环最多用$n(n+1)+2n$.考虑每一个循环和STEP1的所有基本计算次

数，该算法的计算时间复杂性为 $O(n^2)$. 对 0-1 背包问题的任何实例，这个估计都是适用的，因而，这样的估计是从最坏情形分析得到的结果. □

从最坏情形分析来评价一个算法的计算效果时，其评价的指标是计算解的目标值同最优目标值的差距. 这个差距越小说明算法越好. 这样的评价只对启发式算法有效，因为最优算法没有这样的偏差.

记一个实例 I 的最优目标值为 $z_{\mathrm{OPT}}(I)$，启发式算法 H 所得的目标值为 $z_{\mathrm{H}}(I)$. 若优化问题的目标是极大目标函数，对优化问题的任意实例 I，评价函数为

$$d + \alpha z_{\mathrm{OPT}}(I) \leqslant z_{\mathrm{H}}(I) \leqslant z_{\mathrm{OPT}}(I), \tag{1.23}$$

其中，在(1.23)中，$|d| < z_{\mathrm{OPT}}(I)$，$\alpha \leqslant 1$. 对给定的问题和启发式算法 H，我们对问题中任意实例 I 求满足(1.23)式的 α. α 越接近 1 说明构造的启发式算法越好. d 则是一个与实例 I 无关的偏差值. 在常规的研究中，要给出 α 的值并设法验证这个界是否为紧界. 所谓紧界是存在一个实例使(1.23)左边的等号成立.

当优化问题的目标是极小目标函数时，对任意实例 I，评价函数为

$$\alpha z_{\mathrm{OPT}}(I) + d \geqslant z_{\mathrm{H}}(I) \geqslant z_{\mathrm{OPT}}(I) \tag{1.24}$$

其中，在(1.24)中，$|d| < z_{\mathrm{OPT}}(I)$，$d$ 是一个与 I 无关的偏差值，$\alpha \geqslant 1$. 同(1.23)类似，α 越接近 1 说明构造的启发式算法越好. 有时不易确定和验证 α 是紧界，与其接近的方法是确定渐近最坏界，它的定义为

$$\mathrm{AR}(H) = \lim_{k \to \infty} \sup_{I} \left\{ \frac{z_{\mathrm{H}}(I)}{z_{\mathrm{OPT}}(I)} \,\middle|\, z_{\mathrm{OPT}}(I) \geqslant k \right\}. \tag{1.25}$$

通过下面的简单例子说明最坏分析方法.

例 1.15　例 1.10 已给出 0-1 背包问题的贪婪算法，记为 G 算法. 用最坏情形分析该算法，可以构造这样一个例子 I：有两个物

品，$c_1 = 1+\varepsilon, a_1 = 1, c_2 = K, a_2 = K, b = K, \varepsilon$ 为充分小的正数.用例1.10的算法得到 $x_1 = 1, x_2 = 0$，，则 $z_G(I) = 1+\varepsilon$，而 $z_{OPT}(I) = K$. 于是，随着容积 K 的增加，有

$$\frac{z_G(I)}{z_{OPT}(I)} = \frac{1+\varepsilon}{K} \to 0, \quad K \to +\infty. \tag{1.26}$$

任何一个启发式算法，最坏的情形是一个物品也没有装，(1.26) 说明 0-1 背包问题的贪婪算法在最坏情况分析下非常不好，它的渐进效果 $\frac{z_G(I)}{z_{OPT}(I)} \to 0, K \to +\infty$ 表示包中几乎没有装任何物品. □

同样用以上的贪婪算法，应用到一般的背包问题，它的效果则不一样.

例 1.16 一般背包问题

$$z_{OPT}(I) = \max \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \sum_{j=1}^{n} a_j x_j \leqslant b,$$

$$x_j \geqslant 0 \text{ 整数}, j = 1,2,\cdots,n.$$

一般背包问题不同于 0-1 背包问题是一种物品有无限多个，而且可以装多个. 不妨设 $\frac{c_1}{a_1} \geqslant \frac{c_2}{a_2} \geqslant \cdots \geqslant \frac{c_n}{a_n}$，用例 1.10 的贪婪算法，明显的结论是 $x_1 = \left\lfloor \frac{b}{a_1} \right\rfloor, x_2 = \cdots = x_n = 0$ 是一个可行解，其中 $\lfloor x \rfloor$ 表示不超过 x 的最大整数. 所以，

$$z_G(I) \geqslant c_1 \left\lfloor \frac{b}{a_1} \right\rfloor. \tag{1.27}$$

当对模型的决策变量线性松弛时，一般背包问题成为线性规划问题. 因线性规划的最优解在极点达到，而模型仅有容量约束，由线性规划理论，

$$z_{\rm OPT}(I) \leqslant \max\{c_j \frac{b}{a_j} \mid j = 1,2,\cdots,n\} \leqslant c_1 \frac{b}{a_1}. \quad (1.28)$$

比较(1.27)和(1.28)得

$$\frac{z_{\rm G}(I)}{z_{\rm OPT}(I)} \geqslant \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{\frac{b}{a_1}} \geqslant \frac{\left\lfloor \frac{b}{a_1} \right\rfloor}{1 + \left\lfloor \frac{b}{a_1} \right\rfloor} \geqslant \frac{1}{2},$$

所以，$2z_{\rm G}(I) \geqslant z_{\rm OPT}(I)$，$\alpha = \frac{1}{2}$. 下面说明 $\alpha = \frac{1}{2}$ 是渐近紧界. 当

$$c_1 = K + 3\varepsilon,\ c_2 = K,\ a_1 = \frac{K}{2} + \varepsilon,\ a_2 = \frac{K}{2},$$

$$\varepsilon > 0,\ b = K > 2$$

时，贪婪算法 G 得到的解是：装入一个第一类物品，则 $z_{\rm G}(I) = K + 3\varepsilon$. 最优解是装两个第二类物品，$z_{\rm OPT}(I) = 2K$. 所以，

$$\alpha = \lim_{z_{\rm OPT}(I) \to \infty} \frac{z_{\rm H}(I)}{z_{\rm OPT}(I)} = \lim_{K \to \infty} \frac{K + 3\varepsilon}{2K} = \frac{1}{2}$$

为渐近紧界. □

2. 概率分析

最坏情形分析是以最坏的实例来评价一个算法或其解，这样免不了只因一个实例而影响对一个算法或其解的总体评价. 从另一个方面讲，应该从总体来评价算法，概率方法则是从理论上考虑的. 对一个算法同样可以从算法的计算时间复杂性和计算解的效果两个方而进行概率分析. 无论进行那一个方面的分析，都假设实例的数据服从一定的概率分布. 在这个数据概率分布的假设下，研究其算法或解的平均效果.

概率方法在评价算法方面的一个成功应用是对线性规划单纯形法的评价. Klee 和 Minty 在文献[3] 中证明了线性规划的单纯形法不是多项式时间的，其构造的实例为第 2 节中的(1.14). 用概率分析方法研究线性规划问题的一个代表性工作是 Borgwardt[11] 研究线性规划问题

$$\begin{aligned}&\max c^{T}x\\&\text{s.t. } Ax\leqslant e\end{aligned}\qquad \text{其中},A\in\mathbf{R}^{m\times n},e=1\in\mathbf{R}^{m},m\geqslant n$$

所得结果.文献[11]中假设规划的输入数据$c^{T},A_1,A_2,\cdots,A_m$是$\mathbf{R}^n$中具有球面对称测度,得到如下结论:当输入数据在单位球面服从均匀分布,且n充分大时,其迭代次数的渐近上下界分别为$n^{1.5}m^{1/(n-1)}$和$n^2m^{1/(n-1)}$.于是当$n>10$,$m<10^6$时,$m^{1/(n-1)}<4$.当$n>100$,$m<10^6$时,$m^{1/(n-1)}<1.15$.此时,单纯形算法的平均迭代次数为$O(n^{1.5}\sim n^2)$.这说明单纯形算法的平均计算效果是较好的.

概率分析方法是一种理论分析方法,它需要对问题本身有较深入的理解,并且掌握概率模型的建立和概率理论.最坏情形分析和概率分析方法的共同特点是:用理论方法分析算法同最优解之间的误差界,要求有较强的数学基础.

3. 大规模计算分析

前面两种分析方法都是理论分析方法,要求有很强的数学基础和推演能力.正因为如此,目前研究者只对一些特殊和简单的问题采用上面两种理论分析方法.实际中大量的组合优化问题是采用大规模计算分析方法.简单地说,大规模计算分析就是通过大量的实例计算,评价算法的计算效果.算法的计算效果分成两个方面:一方面是算法的计算复杂性,它的效果通过计算机的中央处理器(CPU)的计算时间表现;另一个方面是计算解的性能,它通过计算停止时输出的解表现.

大规模计算分析可对一个算法进行评价.对一个算法进行评价时,它的计算时间效果往往通过目前的计算机设备能否承受、用户能否接受现有的计算时间来衡量.对它的计算解进行评价时,一个简单的要求是用户是否满意,更深一步需要知道问题的最优解或问题的一个下界(假设目标为最小),这时,可以通过数据的计算比较一个算法的计算效果.

大规模计算分析可对多个算法进行评价，比较分析不同算法的效果. 通过大量数据的计算，根据各个算法的计算结果，采用简单或统计的方法比较不同算法的性能. 这种比较不需要已知问题的最优解或上下界.

在大规模数据计算分析过程中，有以下需要注意的因素.

第一个因素是数据的产生. 对实际应用问题，本身存在大量的实际数据，可以用这些数据作为测试对象. 但需要注意的是数据的偏颇性. 因为实际数据往往在某一个限定范围内变化，一旦实际问题出现大的变化，有些算法可能具有不稳定性. 于是，用实际数据评价算法时，可以选出适合这些数据的最好算法. 若实际问题数据有较大的变化时，应该注意算法的适应性. 弥补这方面的不足，还应该用数值实验的方法产生一些有代表性的数据.

研究组合优化问题时，一种常用的方法是产生一些具有代表性的随机数据. 通过这些数据来评价算法的效率. 这些数据的产生应充分考虑代表性. 由此对算法的评价才具有可信性. Soloman[12] 在研究具有时间窗口的车辆线路问题时，给出一个 56 个实例的产生方法. 通过下例可以了解数据产生的一种考虑方法.

例 1.17　经典的具有时间窗口的车辆线路问题可以简单地描述为：m 个车辆服务于 n 个城市，同 n 个城市的 TSP 相同，城市与城市间有路径费用；同时，每个城市有特殊的服务时间称为时间窗口，只有在服务时间内的服务才是有效的；如何安排 m 个车辆使得满足服务时间的要求且费用最小. 随机数据产生的规则如下：

(1) 城市的位置分布

按均匀分布 (uniform)、堆分布 (cluster) 和半堆分布 (semi-cluster) 三类情况. 均匀分布表示城市位置在一个区域内服从均匀分布. 堆分布表示以一些中心点密集分布. 半堆分布则介于均匀分布和堆分布之间. 用图 1.2 表示.

城市位置按上面三种分布产生坐标点.

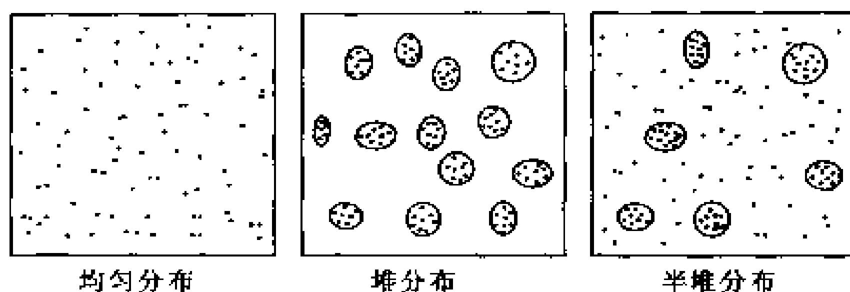


图 1.2 位置分布形式

(2) 时间窗口

时间窗口是指各城市要求服务的开始和完成时间.按 75% 的城市有时间窗口限制和 100% 的城市有时间窗口限制两种要求,时间跨度有大和小两种情况,由均匀分布产生时间窗口数据.

文献[12]共产生 56 个数据实例.这些实例现已成为研究有时间窗口车辆路线问题的典型数据. □

第二个因素是计算结果分析.计算结果分析分为一个算法分析和多个算法比较分析.一个算法结果分析需要知道实例的最优解 $z_{OPT}(I)$ 或下界 $z_{LB}(I)$(目标是极小),算法H目标值为 $z_H(I)$.评价解的性能公式有绝对差

$$z_H(I) - z_{OPT}(I) \text{ 或 } z_H(I) - z_{LB}(I). \tag{1.29}$$

相对差

$$\frac{z_H(I) - z_{OPT}(I)}{z_{OPT}(I)} \text{ 或 } \frac{z_H(I) - z_{LB}(I)}{z_{LB}(I)}, \tag{1.30}$$

或

$$\frac{z_H(I) - z_{OPT}(I)}{z_H(I)} \text{ 或 } \frac{z_H(I) - z_{LB}(I)}{z_H(I)}. \tag{1.31}$$

通过采用上述的公式,最终可采用典型数据或用统计处理的方法,比较算法的性能.需要注意评价公式中,应尽量根据已有的信息以避免过多的计算.如在(1.30)和(1.31)各个评价公式中,

每个公式用到两个计算值，而没有同时采用三个计算值.

对于解的稳定性分析，可采用统计学中的方差

$$D = \sum_{I \in S} (z_H(I) - \bar{H})^2, \tag{1.32}$$

其中

$$\bar{H} = \frac{1}{|S|} \sum_{I \in S} z_H(I), \tag{1.33}$$

S 表示所有数据实例的集合，$|S|$ 表示 S 中包含的实例数.

算法所耗用的计算时间也是一个重要的指标.

对多个算法的分析比较时，两种算法 H_1 和 H_2，若优化问题的目标是极小目标函数，对任意的实例 I

$$z_{H_1}(I) < (\leqslant) z_{H_2}(I), \tag{1.34}$$

则称 H_1 优于（不次于）H_2.

多个算法的分析比较同样可以考虑解的性能、解的稳定性和耗用时间等指标. 可以用(1.32)的方差公式考虑耗用时间的稳定性. (1.34)式是对每一个实例进行比较，同样，可用该公式比较一组数据的平均效果. 类似(1.34)，可以比较稳定性、计算时间等指标.

最后，用图 1.3 表示启发式算法解之间的关系. 对极小目标函数的优化问题，理论上可以保证并且很容易证明，任何一个求解优化问题可行解的启发式算法的目标值都是最优值的上界. 在图1.3中，一步法、改进法、分支定界启发式、割平面启发式、线性规划松弛再对解可行化、拉格朗日松弛可行化、禁忌搜索、模拟退火、遗传算法、人工神经网络、限制解空间、分解法和组合算法等的目标值都是最优值的上界. 理论上无法区分这些算法的目标值之间的顺序关系，要了解这些算法对一个问题的适用性，只有通过大规模数值计算分析比较. 线性规划松弛和拉格朗日松弛等算法则提供优化问题的下界，如图 1.3 所示，它们的目标值不超过最优值.

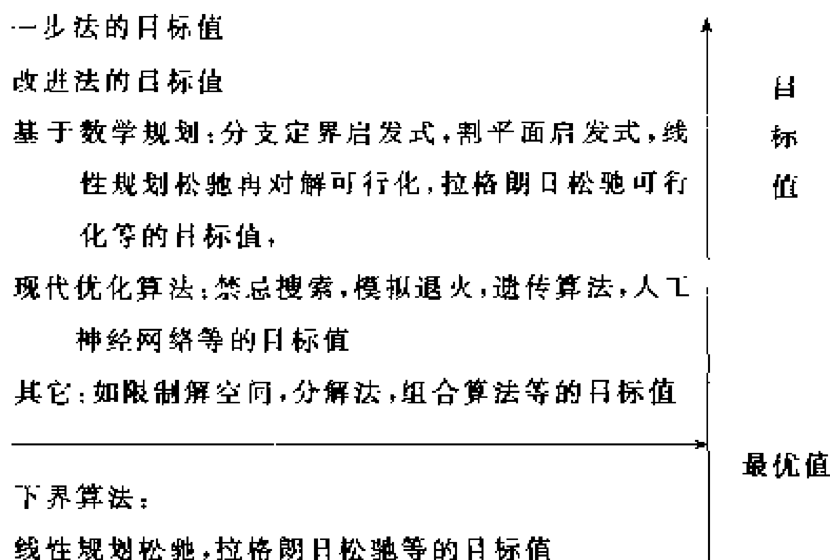


图 1.3 启发式算法目标值的关系

1.5 NP,NP-C 和 NP-hard 概念

第 2 节初步介绍了计算复杂性的一些概念，我们知道了多项式问题，同时也知道像例 1.1 ～ 1.5 的组合优化问题还没有找到多项式时间的最优算法. 本节将以组合最优化问题为研究对象，介绍计算复杂性的 NP、NP-C 和 NP-hard 概念. 通过几个典型组合优化问题一步一步地展开、深入，逐渐理解有关 NP、NP-C 和 NP-hard 的概念. 同时，以这些问题的一步步深入，介绍如何研究一个组合优化问题的复杂性.

NP，NP-C 和 NP-hard 复杂性问题的研究涉及计算机概念. 在此，仅用组合优化中较为直观和通用的语言简单描述.

首先，我们介绍一些有关组合优化问题复杂性的基本概念. 复杂性的研究是从“问题(problem)”开始的. 问题是需要回答的一般性提问，通常含有若干个满足一定条件的参数. 问题通过下面的描述给定：

(1) 描述所有参数的特性.

(2) 描述答案所满足的条件. 一旦所有的参数确定,则我们称之为问题的一个实例.

定义 1.6 给定一个组合优化问题,当问题中的参数赋予具体值时,称为问题的一个实例(instance). 这些具体值称为数据,这些数据输入计算机所占空间称为实例的输入长度(size).

于是,对问题、实例和输入长度三个概念,我们用表 1.2 和表 1.3 所列内容来理解.

表 1.2 问题和实例

问题	实例
TSP	例 1.2 问题中的各参数为:100 个城市,城市间距离 d_{ij} 已知.
背包问题	例 1.1 问题中的各个参数为:4 个物品,大小分别为 4,3,2,2. 价值分别为 8,7,5,7. 包的大小为 6.
整数线性规划	例 1.3 问题中的 n, A, b, c 已知.

实例是问题的特殊表现,只有给定了数据,才能通过计算机解决. 实例的输入长度同计算机中的存储编码方式有关,如采用二进制编码,则通过表 1.3 了解输入长度的概念.

表 1.3 编码与输入长度

数据	二进制	输入长度
1	1	1
2	10	2
10	1010	4
64	1000000	7
$k \geqslant 2$		$\leqslant \lceil \log_2 k \rceil + 1$

一般以正整数$\beta(\beta > 1)$进制记录一个正整数n的展开形式为

$$n = a_k\beta^k + a_{k-1}\beta^{k-1} + \cdots + a_0\beta^0, \tag{1.35}$$

其中,$a_k \neq 0$,它的β进制数据为$(a_k a_{k-1} \cdots a_0)$. 由于

$$a_k\beta^k \leqslant \beta^{\log_\beta n} = n \leqslant \beta^{k+1} \leqslant \beta^{\log_\beta n + 1}$$

输入长度$k+1$在$\log_\beta n$与$\log_\beta n + 1$之间,或用整数$\lceil \log_\beta n \rceil + 1$估计输入长度的上限. 对多个数据的实例,采用数据累计的方法记录实例的输入长度. 例1.6线性规划问题实例的输入长度的估计就是采用数据累计的方法. 由于计算机中只有有限位有效数,故所有数据都以有理数讨论. 由于整数0的输入长度为1,定义$\log_\beta 0 = 0$后,上面的结论可以推广到非负整数.

注:此处都是对非负整数考虑输入长度,即没有考虑实际数据中的负号. 以例1.6的线性规划为例,它的很多系数可以有正负号的,约束中还出现$\leqslant$, $\geqslant$和$=$等方程,这些都没有在上面讨论的实例输入长度中考虑. 它们将在下面讨论的算法中体现.

一个实例完全由它的数据决定. 给定一个问题的数据,即给定一个实例,我们可以用一种已知的计算机上的方法去求解. 这种计算机上的求解方法称为算法. 算法不仅仅局限于每一个实例,而是求解问题的通用程序. 评价算法的一个主要指标是计算所耗用的时间. 耗用的时间可以通过两种方法计量. 第一种计量方法是计算机中央处理器处理实例的工作时间. 由于计算时间可能因计算机的运算速度不同而产生差别,这种计量方法因计算机的不同而失去了比较的效能. 另一种计量方法是估计算法的基本运算总次数,即算法中加、减、乘、除和比较等的总次数. 无论如何,算法的加、减、乘、除和比较等基本运算是确定的,于是,一个算法基本运算的总次数也就称为该算法的计算时间. 衡量一个算法的计算效率应该考虑计算时间和其计算实例的输入长度. 一个算法解决一个问题,我们以此为目的构造算法. 在第2节中我们已介绍:一个算法

的功效是通过该算法的最坏实例的计算时间来评价；一个问题是否为多项式问题是通过研究是否存在一个最优算法和一个多项式函数 $g(x)$ 使得(1.13)对所有的实例成立.

迄今为止，对许多的组合优化问题都没有找到求最优解的多项式时间算法，因此，比多项式问题更广泛的一类问题是非多项式确定(nondeterministic polynomial，简记 NP)问题. NP 的概念是由判定问题引入的.

定义 1.7 如果一个问题的每一个实例只有“是”或“否”两种答案，则称这个问题为判定问题. 称有肯定答案的实例为“是”实例. 称答案为“否”的实例为“否”实例或非“是”实例.

在研究组合优化问题复杂性时，处理的方法是对给定的一类优化问题，将其转化为判定问题，使对每一个实例只有“是”或“否”的回答. 例 1.18 和 1.19 就将组合优化问题转化为判定问题.

例 1.18 例 1.6 的线性规划问题(LP)转化为一个判定问题——LP 判定问题：

先给出一个目标值 z，将极小化目标函数转化为：判定是否有可行解使其目标值不超过这个 z. 判定问题的数学表达式为：

$$\text{给定 } z\text{，是否有}\{\boldsymbol{x}\,|\,\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\leqslant z,\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b},\boldsymbol{x}\geqslant\boldsymbol{0}\}\neq\varnothing?$$

$\boldsymbol{x}\in\{\boldsymbol{x}\,|\,\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\leqslant z,\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b},\boldsymbol{x}\geqslant\boldsymbol{0}\}$ 称为对应线性规划判定问题的可行解. 同样，整数线性规划也可以写成同上类似的判定问题. □

例 1.19 例 1.2 的 TSP 问题转化为一个判定问题.

用 n 个城市的一个排列 $(i_1,i_2,\cdots,i_n)$ 表示商人从城市 i_1 出发依次通过 $i_2,\cdots,i_n$，最后返回 i_1 这样一个路径. 判定问题为：给定 z，是否存在 n 个城市的一个排列 $\boldsymbol{W}=(i_1,i_2,\cdots,i_n)$，使得

$$f(\boldsymbol{W})=\sum_{j=1}^{n}d_{i_j i_{j+1}}\leqslant z?$$

其中 $i_{n+1}=i_1$. 满足 $f(\boldsymbol{W})=\sum_{j=1}^{n}d_{i_j i_{j+1}}\leqslant z$ 的一个排列 $\boldsymbol{W}$ 称为对应

判定问题的一个可行解. □

如下面的例 1.20 和例 1.21 所示,有些组合问题本身是一个判定问题.

例 1.20 适定性问题.

在逻辑运算中,布尔变量 x 的取值只有两个:“真”和“假”,逻辑运算有“或(+)”,“与(·)”和“非($^{-}$)”.如果“真”用“1”表示,“假”用“0”表示,布尔运算遵循表 1.4(a)(b)(c).

表 1.4(a) 运算“+”

布尔变量值		运算结果
0	0	0
0	1	1
1	0	1
1	1	1

表 1.4(b) 运算“·”

布尔变量值		运算结果
0	0	0
0	1	0
1	0	0
1	1	1

表 1.4(c) 运算“$^{-}$”

布尔变量值	运算结果
0	1
1	0

设$\{x_1,x_2,\cdots,x_n\}$为 n 个布尔变量,布尔变量及其“非”组成的集合$\{x_1,x_2,\cdots,x_n;\overline{x_1},\overline{x_2},\cdots,\overline{x_n}\}$称为文字集.文字集的任意一个子集及其各元素的“或”运算组成一个句子.m 个句子及之间的“与”运算组成一个表达式.

如表达式

$$(x_1+\overline{x_2})\cdot x_2$$

的两个句子为 $x_1+\overline{x_2}$ 和 x_2.当每一个逻辑变量取一定值时,采用逻辑运算关系,可以计算布尔表达式的值.在上式中,当 $x_1=1,x_2=1$ 时,表达式为“真”.一组使表达式为“真”的变量取值称为真值分配.存在真值分配的表达式称为适定的.容易证明,表达式

$$(x_1+\overline{x_2})\cdot(\overline{x_1}+x_2)\cdot(x_1+x_2)\cdot(\overline{x_1}+\overline{x_2})$$

就不是适定的.

适定性问题定义为:

给定包含 n 个布尔变量的 m 个句子 $C_1,C_2,\cdots,C_m$,布尔表达式

$C_1 \cdot C_2 \cdot \cdots \cdot C_m$ 是适定的吗？ □

例 1.21　三精确覆盖.

已知 $S=\{u_1,u_2,\cdots,u_{3m}\}$ 的 n 个子集构成的子集族 $F=\{S_1,S_2,\cdots,S_n\}$，其中每个子集包含 S 中三个元素，F 中是否存在 m 个子集 $S_{i_1},S_{i_2},\cdots,S_{i_m}$，使得 $\bigcup_{j=1}^{m} S_{i_j} \supset S$？$m$ 个子集 $S_{i_1},S_{i_2},\cdots,S_{i_m}$ 满足 $\bigcup_{j=1}^{m} S_{i_j} \supset S$，称为 m 个子集 $S_{i_1},S_{i_2},\cdots,S_{i_m}$ 覆盖 S. □

对于判定问题，我们可以将求解该问题的算法分为两个阶段. 第一阶段称为猜测阶段. 它的主要工作是求解或猜测该问题的一个解. 对于“是”实例，我们的目标是能够找到回答“是”实例的解. 第二个阶段称为检查或验证阶段. 一旦解已经选定，将猜测的解作为输入，验证此解是否为该实例“是”的回答. 我们称实例回答“是”的解为实例的可行解，否则称为不可行解. 确定一个实例是否存在回答“是”的解是难度较大的工作，它需要从众多的解中寻找.

时常采用“解”表示问题的一个解决方案. 在计算机计算的过程中，问题的每一个实例在计算机输入时都以一定结构的字符串表示，如例 1.6 的线性规划问题可以用一个长度为$(mn+m+n)(1+\log_2|p|)$ 的 01 字符串表示. 实例的每一个解也同样可以用一个字符串表示. 在计算的过程中，由于求解判定问题的两阶段性，第一阶段涉及实例的输入，因此，有实例的输入长度. 第二阶段以第一阶段的猜测解作为输入，因此，产生猜测解的输入长度. 在此，为了不引起混淆，在输入长度前分别强调实例输入长度和猜测解输入长度.

定义 1.8　若存在一个多项式函数 $g(x)$ 和一个验证算法 H，对一类判定问题 A 的任何一个“是”的判定实例 I，都存在一个字符串 S 是 I 的“是”答案，满足其输入长度 $d(S)$ 不超过 $g(d(I))$，其中 $d(I)$ 为 I 的输入长度，且算法 H 验证 S 为实例 I 的“是”答案的计算时间 $f(H)$ 不超过 $g(d(I))$，则称判定问题 A 是非多项式确

定的，简记为NP. 在不混淆的情况下，用NP表示所有非多项式确定问题的集合.

由上面的定义知，判定问题是否属于NP的关键是对“是”的判定实例，“存在”一个字符串和算法，满足以上定义. 我们不须回答“存在”的算法和字符串是怎样得到的，这里字符串的含义可以理解为实例的一个解. 定义1.8中两处用到多项式的限制，一处是“是”答案S的输入长度不超过$g(d(I))$，另一处是算法H验证S为实例I的“是”答案的计算时间不超过$g(d(I))$. 可以将上面定义中这两处的多项式限制改变为：存在两个多项式函数$g_1(x)$和$g_2(x)$，使得“是”答案S的输入长度不超过$g_1(d(I))$，算法H验证S为实例I的“是”答案的计算时间不超过$g_2(d(I))$. 因为只要选高阶的多项式作为定义1.8的$g(x)$，这样的定义与原定义1.8等价.

对非“是”实例，我们不讨论它的相关判定的结果. 由定义可知$\mathrm{P}\subset\mathrm{NP}$. 用下面例1.22～1.24理解如何确定一个判定问题是否属于NP.

例1.22 （续1.18）有理系数的LP判定问题属于NP.

例1.18已给出LP判定问题. 由于LP的解可由n个变量的取值决定，任何一个基本可行解的各分量满足下面性质. 下面用到线性规划的基本结论，主要包括：极点的概念、基本可行解的性质等，相关的内容可参考线性规划方面的论著（如文献[13]）.

性质1.1 LP任何一个基本可行解的各分量及目标函数值$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$有界，且其二进制字符串表达式的输入长度不超过$L=mn+m+n+\log_2|p|$，即基本可行解的输入长度不超过实例输入长度的多项式函数，其中p在例1.6中定义.

证明 按假设LP的系数为有理数和$m\leqslant n$，因此等价为LP的所有系数为整数. 对LP的任何一个基本可行解，当分量$x_j=0$时，性质结论显然成立. 当基变量分量$x_j\neq 0$时，由线性规划的结

论，$x_B = B^{-1}b$，其中 B 为基矩阵. 再由代数的基本结论 $B^{-1} = B^*/\det(B)$，其中 B^* 为 B 的伴随矩阵. 在没有简约的情况下，知任何一个非零基变量的分母为 $\det(B)$，分子为 $\sum_{i=1}^{m} b_i A_{ij}$，其中 A_{ij} 为 B 第 i 行 j 列的代数余子式. 于是分子、分母和目标函数值分别满足

$$1 \leqslant |\det(B)| \leqslant m!|p|,$$

$$\left|\sum_{i=1}^{m} b_i A_{ij}\right| \leqslant m!|p|,$$

$$\left|\sum_{j=1}^{n} c_j x_j\right| = \left|\sum_{j=1}^{m} (c_j \sum_{i=1}^{m} b_i A_{ij})\right| \leqslant (m+1)!|p|.$$

由于 $m \leqslant n, m < m+1 \leqslant 2^m \leqslant 2^n$ 和 $n < 2^n$，得到目标值和基本可行解的每一个分量值不超过

$$m!|p| < (m+1)!|p| \leqslant 2^{n(m+1)}|p| < 2^{mn+m+n}2^{\log_2|p|},$$

所以，基本可行解的任何一个分量和目标值的输入长度都不超过

$$L = \log_2 2^{mn+m+n+\log_2|p|}.$$

由例 1.6 的讨论，实例的输入长度 $d(I)$ 的下限是 $\max\{mn+m+n, \log_2|p|\}$. 再由上面的讨论，基本可行解的输入长度不超过 nL，只需取多项式函数 $g(x) = 2x^2$，则有，基本可行解的输入长度不超过 $g(d(I))$. 性质成立. □

对 LP 判定问题的一个"是"实例，由线性规划的结论，存在一个基本可行解. 当给定 LP 的一个基本可行解 x，只要验证是否满足 $\{c^T x \leqslant z, Ax \geqslant b, x \geqslant 0\}$？验证时最多需 $L_1 = (m+1)n$ 个乘，$L_2 = (m+1)(n-1)$ 个加或减，$L_3 = m+1$ 个比较. 当 $m, n \geqslant 1$ 时，如此设计的验证算法计算时间不超过 $L_1 + L_2 + L_3 = 2mn + 2n < 2\max\{mn+m+n, \log_2|p|\}$. 基本可行解的输入长度不超过 nL，满足 $nL < 2\max\{mn+m+n, \log_2|p|\}^2$. 由例 1.6 知实例的输入长度的下限为 $\max\{mn+m+n, \log_2|p|\}$，故定义 1.8 中的多项式函数可以选 $g(x) = 2x^2$. 此时，已有多项式函数 $g(x) = 2x^2$

和简单的验证算法，使得验证一个基本可行解是否为 LP 判定问题的一个可行解所花费的时间和基本可行解的输入长度满足定义 1.8，所以 LP ∈ NP. □

例 1.23 （续例 1.19）TSP 问题属于 NP.

例 1.19 给出了对应的判定问题. 对任何一个 n 城市 TSP 实例，它的解形式是 $(1,2,\cdots,n)$ 的一个排列 $\boldsymbol{W}=(i_1,i_2,\cdots,i_n)$，显然该字符串输入长度不超过 $\sum_{i=1}^{n}\left(\lceil \log_2 i \rceil + 1\right) < n + n\lceil \log_2 n \rceil$. 实例的输入长度可由城市间的距离数据确定，城市间的距离数据有 $n(n-1)$ 个（不考虑城市自身间的距离），城市 i 和 j 之间的一个非零距离 d_{ij} 的输入长度不低于 $\max\{1,\log_2|d_{ij}|\}$，因此，实例的输入长度不低于 $L=\max\{n(n-1),\log_2|P|\}$，其中，$P=\Pi\{d_{ij}\mid d_{ij}\neq 0\}$. 这样估计出实例的输入长度不低于 $n(n-1)$. 上面的讨论表明，TSP 任何一个解的输入长度不超过 $n\left(1+\lceil \log_2 n \rceil\right) \leqslant n(n-1)+4$，不超过实例输入长度 $L+4$.

若给定 TSP 判定实例的一个可行解，只需验证

$$f(\boldsymbol{W}) = \sum_{j=1}^{n} d_{i_j i_{j+1}} \leqslant z?$$

验证算法有 n 个加法和一个比较，算法的计算时间为 $n+1<L+4$. 于是，定义 1.8 中的多项式函数可选 $g(x)=x+4$，验证算法的计算时间和判定问题实例可行解的输入长度都满足定义 1.8 的要求，故 TSP ∈ NP. □

例 1.24 （续例 1.20）三精确覆盖属于 NP.

S 集合中共有 $3m$ 个元素，按下标排列这些元素，此时，子集族 F 中的每一个元素 $S_i(i=1,2,\cdots,n)$ 对应一个 $3m$ 维向量. 向量形式为：向量的 $3m$ 个分量分别对应 $3m$ 个元素，包含在对应子集中的元素为 1，余下的为 0. 例如，6 个元素 $S=\{u_1,u_2,\cdots,u_6\}$，F 中的一个元素（即 S 的一个子集）$S_1=\{u_1,u_2,u_6\}$，对应向量为 $\boldsymbol{\alpha}_1=(1,$

1,0,0,0,1). 表示为一个字符串(110001). F 中 m 个元素 $\{S_{i_1}, S_{i_2}, \cdots, S_{i_m}\}$ 是 F 的一个精确三覆盖的充分必要条件为字符串向量问题满足:相应 m 个字符串对应的向量 $\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_m}$ 满足 $\sum_{j=1}^{m} \alpha_{i_j} = (1,1,\cdots,1)$.

无论如何,三精确覆盖问题任何一个实例的输入长度至少需要占用长度 $3m$. 上面的讨论得到,三精确覆盖问题转化为一个字符串向量问题. 按字符串向量问题,精确三覆盖任何一个"是"实例的解可以用输入长度为 $3m^2$ 的字符表示. 验证 $\sum_{j=1}^{m} \alpha_{i_j} = (1,1,\cdots,1)$ 是否成立的算法需 $3m^2$ 个加法和 $3m$ 个比较,算法的计算时间为 $3m^2 + 3m$. 多项式函数可选 $g(x) = 3x^2 + x$,则定义 1.8 条件满足,所以三精确覆盖属于 NP. □

研究中常常有这样一种思路,我们设法将几类问题归结为一个问题,一旦解决了这一个归结后的问题,其他的几类问题也就得到解决. 我们称这种方法为归类或归约. 处理判定问题也有类似的方法.

定义 1.9 给定问题 $A1$ 和 $A2$,若存在一个多项式函数 $g(x)$ 和一个字符串映射关系满足:

(1) 对 $A1$ 的任何一个实例 I_1,在实例 I_1 输入长度的多项式时间 $g(d(I_1))$ 内构造 $A2$ 的一个实例 I_2,使其输入长度不超过 $g(d(I_1))$(其中 $d(I_1)$ 为实例 I_1 的输入长度).

(2) 由此构造使得实例 I_1 和 I_2 的解一一对应,且 d_1 为 I_1 的"是"答案的充分必要条件为 d_1 对应的解是 I_2 的一个"是"答案,则称问题 $A1$ 多项式转换为(Transformation)$A2$ 问题.

用下面例 1.25 和例 1.26 理解定义 1.9.

例 1.25 (续例 1.20) 适定性问题多项式转换为整数线性规划问题.

将适定性问题的逻辑变量"真"对应"1","假"对应"0",于是

逻辑变量 x 对应整数变量,$\bar{x}$ 对应 $1-x$. 一个 n 个变量的句子 $C_j(j=1,2,\cdots,m)$ 要求是"真",对应下列不等式是否有可行解?

$$\sum_{x_i \in C_j} x_i + \sum_{\bar{x}_i \in C_j} (1 - x_i) \geqslant 1, \quad j = 1,2,\cdots,m, \qquad (1.36)$$

$$x_i \in \{0,1\}, \quad j = 1,2,\cdots,n. \qquad (1.37)$$

对于适定问题的任何一个实例,它是由 n 个逻辑变量和 m 个句子组成,按实例的定义,m 个句子中的逻辑变量值是已知的数据,故适定性问题实例的输入长度是 mn. 由(1.36)和(1.37)的映射,对应 m 组(1.36)和(1.37)形式的不等式是整数线性规划问题,对应整数线性规划实例的输入长度按例 1.6 的估计不超过 $(m+n)m+(m+n)+m+m\log_2 n$. 因此,定义 1.9 中的多项式函数选 $g(x)=6x^2$,就有定义 1.9 的(1)成立. 易证适定问题转换到对应(1.36)和(1.37)约束的整数规划问题满足(2),说明适定性问题多项式转换为整数线性规划问题. □

研究例 1.1 的 0-1 背包问题的一种特殊情形 $c_j=a_j, j=1,2,\cdots,n$,可以给出一个判定问题:0-1 背包判定问题

对给定的整数 $c_j, j=1,2,\cdots,n$ 和 b,是否存在 $\{1,2,\cdots,n\}$ 的子集 B,使得 $\sum_{j\in B} c_j = b$?

例 1.26 三精确覆盖多项式转换为 0-1 背包判定问题.

由例 1.24 的 S_j 与二进制向量对应关系,一个二进制向量又可一一与一个整数对应:

$$S_j \longrightarrow c_j = \sum_{u_i \in S_j} (n+1)^{i-1} \qquad (1.38)$$

如此,将一个向量一一对应于一个整数. 由再对应于 $3m$ 个分量全为 1 的向量 $(1 \quad 1 \quad \cdots \quad 1)$,选 K 为

$$(1 \quad 1 \quad \cdots \quad 1) \rightarrow K = \sum_{j=1}^{3m} (n+1)^{j-1}. \qquad (1.39)$$

容易验证上面对应满足定义 1.9(1). 由上面的构造,三精确覆盖

实例的一个解$\{S_{i_1},S_{i_2},\cdots,S_{i_m}\}\in F$——对应背包问题实例的一个解$B=\{i_1,i_2,\cdots,i_m\}$. 此时,$S$存在一个精确三覆盖$\{S_{j_1'},S_{j_2'},\cdots,S_{j_m'}\}$的充要条件为$\sum_{i=1}^{m}c_{j_i'}=K$.

充分性：假设存在$T\subseteq\{1,2,\cdots,n\}$，使得$\sum_{j\in T}c_j=K$. 由(1.38)和(1.39),$K$的每个指数项系数为1. 又因为$T$的总项数不超过$n$,其同次幂项的系数和不超过$n$,因而,由$T$和(1.39),$n+1$的每个次幂项系数恰好为1. 由(1.38)的反变换,得$S_j,j\in T$是S的一个三精确覆盖.

必要性：由(1.38)和(1.39)的定义,可得结论.

由上面的讨论,映射关系(1.38)和(1.39)满足定义1.9(2),所以,三精确覆盖多项式时间转换0-1背包判定问题. □

Cook[14]在1971年给出并证明了有一类问题具有下述性质：(1)这类问题中任何一个问题至今未找到多项式时间算法；(2)如果这类问题中存在一个问题有多项式时间算法,那么这类问题都有多项式时间的算法，这类问题记为NPC或NP-C(NP-Complete). 下面给出定义.

定义1.10　已知判定问题$A1$和$A2$,若存在多项式函数$g_1(x)$和$g_2(x)$,使得对$A1$的任何实例I,在多项式时间内构造$A2$的一个实例,其输入长度不超过$g_1(d(I))$,并对$A2$的任何一个算法H2,都存在问题$A1$的一个算法H1,使得H1算法调用H2算法并使得计算时间$f_{\mathrm{H1}}(d(I))\leqslant g_2(f_{\mathrm{H2}}(g_1(d(I))))$,其中$f_{\mathrm{H1}}(d(I))$和$f_{\mathrm{H2}}(d(I))$分别表示算法H1和H2的计算时间与实例输入长度$d(I)$之间的关系,则称问题$A1$多项式归约(reduction)为$A2$问题.

从定义1.10看出,若$A2$存在多项式时间算法,则$A1$一定存在多项式时间算法,比较定义1.9和定义1.10,若问题$A1$多项式

转换为问题 $A2$，对 $A2$ 的一个算法 H2，构造求解 $A1$ 问题的算法可遵循如下步骤：对问题 $A1$ 的任何一个实例 I_1，先用多项式时间 $g_1(d(I_1))$ 构造 $A2$ 问题的一个实例 I_2；再调用算法 H2 求解 I_2，这一步的计算时间为 $f_{H2}(g_1(d(I_1)))$. 这样构造的算法对 $A1$ 问题任意实例 I_1 的计算时间不超过 $g_1(d(I_1))+f_{H2}(g_1(d(I_1)))$，满足定义 1.10，$A1$ 多项式归约为 $A2$ 问题. 由此得到定义 1.10 比定义1.9更广泛.

用例 1.27 来理解定义 1.10.

例 1.27　（续例 1.20）适定问题多项式归约为三精确覆盖问题.

例 1.20 定义的适定性问题为：$x_1,x_2,\cdots,x_n$ 为逻辑变量，$C_1,C_2,\cdots,C_m$ 为句子. 现在构造一个三精确覆盖问题，共分三个部分构造三精确覆盖的元素集合 S.

第一部分对所有的逻辑变量及其非进行复制，对应每一句子复制一次，共 $2mn$ 项，记成

$$U=\{x_i^j,\overline{x_i^j}\mid i=1,2,\cdots,n;\ j=1,2,\cdots,m\}.$$

第二部分包含三种类型的元素共 $2mn$ 项.

$$V=\{a_i^j\mid i=1,2,\cdots,n;j=1,2,\cdots,m\}\cup\{v_j\mid j=1,2,\cdots,m\}$$
$$\cup\{c_i^j\mid i=1,2,\cdots,n-1;j=1,2,\cdots,m\}.$$

第三部分同第二部分的构造基本相同，

$$W=\{b_i^j\mid i=1,2,\cdots,n;j=1,2,\cdots,m\}\cup\{w_j\mid j=1,2,\cdots,m\}$$
$$\cup\{d_i^j\mid i=1,2,\cdots,n-1;j=1,2,\cdots,m\}.$$

于是，S 中共包含 $6mn$ 个元素. 三元素子集族 F 仅由下列三部分子集族组成：

第一部分：由 U,V,W 中各选一个元素，组成 (x_i^j,a_i^j,b_i^j)，$i=1,2,\cdots,n;j=1,2,\cdots,m$ 或 $(\overline{x_i^j},a_i^{j+1},b_i^j)$，$i=1,2,\cdots,n;j=1,2,\cdots,m$，其中，定义 $a_i^{m+1}=a_i^1$. F 中包含 a 和 b 类型的元素仅有上面

这些组合形式，则对任意 i，任何覆盖 $a_i^j, b_i^j, i=1,2,\cdots,n; j=1,2,\cdots,m$ 的所有三元素子集族中必是同由 a 和 b 类型的元素与 $x_i^j, j=1,2,\cdots,m$ 组合，即 $(x_i^j, a_i^j, b_i^j), j=1,2,\cdots,m$，或是同与 $\overline{x_i^j}, j=1,2,\cdots,m$ 组合，即 $(\overline{x_i^j}, a_i^{j+1}, b_i^j), j=1,2,\cdots,m$. 这一部分包含 $2mn$ 个元素.

第二部分：由 $(y_j, v_j, w_j), j=1,2,\cdots,m$ 组成，其中，y_j 是句子 C_j 中的一个逻辑变量. 由于 U 包含所有逻辑变量的"是"及其"否"，自然包含 y_j. 因此，第二部分同第一部分仅在这一位置可能出现相同元素. 由 F 的第一、第二部分组成，知 $y_j \in C_j \cap U$. 这一部分包含 m 个元素.

第三部分：$(y_k, c_i^j, d_i^j), i=1,2,\cdots,n-1; j=1,2,\cdots,m, k=1,2,\cdots,m$，其中，$y_k \in U$. 这一部分包含 $(n-1)mm$ 个元素.

三元子集族 F 中共有 $m^2(n-1)+m+2mn$ 个子集.

假设适定性问题的一个实例由 n 个逻辑变量的 m 个句子组成，实例的输入长度由 m 个句子中的逻辑变量值确定，因此实例的输入长度为 mn. 上面已构造三精确覆盖问题的一个特殊情况，这是一种映射，简单估计三精确覆盖问题实例的输入长度不超过 $|S||F|$，其中 $|S|$ 和 $|F|$ 分别表示集合 S 和 F 中元素的个数，即不超过 $6mn(m+2mn+m^2n)$，定义 1.10 中的多项式函数选为 $g_1(x)=6x^3+18x^2$.

当三元素族 F 中存在 S 的一个三精确覆盖实例时，必包含第二部分的 m 组，取三精确覆盖中第二部分 y_j 的逻辑变量为"真"值，即当 $y_j = x \in C_j$ 时，$x=1$；$y_j=\overline{x} \in C_j$ 时，$x=0$. 则对应适定性问题的一个"是"实例. 反之，对适定问题的任何一个"是"实例，按上面的构造，则对应三精确覆盖也是一个"是"实例，其三精确覆盖是：在 F 集合中的第二部分 $(y_j, v_j, w_j), j=1,2,\cdots,m$ 中，取 y_j 为 C_j 中取值为"真"的逻辑变量；在 C_j 中逻辑变量 y_j 可能有

两种形式 $y_j = x_i$ 或 $y_j = \overline{x_i}$，以 $y_j = x_i = 1$ 的情况来讨论(同法讨论 $y_j = \overline{x_i} = 1$ 的情况)：在 F 的第一部分中，取 $\overline{x_i^j}(j = 1,2,\cdots,m)$ 对应的所有三元组．至此，共选取 $(n+1)m$ 个三元组；最后，在 F 的第三部分中，选取覆盖 S 中 U,V,W 的余下部分的 $(n-1)m$ 个三元组．由此，上面构造的问题可用三精确覆盖问题的一个算法求解，可令 $g_2(x) = x$．因此，适定问题可由多项式归约为三精确覆盖问题． □

定义 1.11 判定问题 $A \in$ NPC(NP-complete)，其定义为：$A \in$ NP 且 NP 中的任何一个问题可在多项式时间内归约为 A．若 NP 中的任何一个问题可在多项式时间归约为判定问题 A，则称 A 为 NP 难或 NP-hard．

于是有 NPC $\subset$ NP-hard．NPC 和 NP-hard 两者的区别是 NP-hard 问题无须判断 A 是否属于 NP．验证一个问题 A 是否为 NPC 的关键有两点，一是 NP 中的任何一个问题是否可在多项式时间内归约为 A；其次，是否存在一个字符串，其输入长度为实例输入长度的多项式函数，和一个多项式时间的验证算法．

由多项式转换、NPC、NP-hard 等的定义，当已知一个问题为 NPC 或 NP-hard，再遇到一个新问题时，只要已知问题可由多项式归约为新问题，则知新问题是 NP-hard．当可以验证新问题是属于 NP 时，则该问题属于 NPC．由此可知，研究一个新问题的复杂性，往往借助于已知问题的复杂性和多项式归约．通过下面例子可以了解这一过程．

例 1.28 适定问题是 NPC．

证明可见文献[14]． □

我们认为该问题是已知的．下面是一种比较典型的研究方法，如果可能的话，将适定问题归约到所研究的问题．问题复杂性也就由此而得到．

例 1.29 整数线性规划的判定问题属于 NPC．

按例 1.18 的说明，整数线性规划的判定问题类似 LP 的判定问题. 例 1.22 证明 LP 的判定问题属于 NP，整数线性规划的判定问题是 LP 判定问题的一个子类，故知整数线性规划的判定问题属于 NP. 例 1.25 已证明适定问题归约为整数线性规划的判定问题，所以，整数线性规划的判定问题是 NPC. □

例 1.30　三精确覆盖属于 NPC.

由例 1.24 和例 1.27 得此结论. □

例 1.31　背包判定问题属于 NPC.

背包问题的解对应一个 n 维 0-1 分量的向量，于是，很易验证属于 NP. 例 1.26 已证明三精确覆盖多项式转换为背包判定问题，所以，背包判定问题属于 NPC. □

背包判定问题的一个特殊情况，包的容积 $K=\frac{1}{2}\sum_{j=1}^{n}c_j$，此时，对应一类新的问题 —— 集合划分(partition) 问题：

给定整数 $c_1,c_2,\cdots,c_n$，是否存在 $\{1,2,\cdots,n\}$ 的一个子集 S，使得 $\sum_{j\in S}c_j=\sum_{j\notin S}c_j$?

例 1.32　集合划分问题是 NPC.

因背包问题属于 NP，所以，集合划分问题属于 NP. 例 1.31 已证明背包判定问题属于 NPC，只须证明，背包问题可以多项式转换为集合划分问题.

对给定的 0-1 背包判定问题的任何一个实例，$c_1,c_2,\cdots,c_n,K$，构造集合分划问题的实例 $c_1,c_2,\cdots,c_n,c_{n+1}=2M,c_{n+2}=3M-2K$，其中，$M=\sum_{j=1}^{n}c_j>K$.

显然，这个映射满足定义 1.9 的(1). 存在$(1,2,\cdots,n)$ 的子集 S，使得 $\sum_{j\in S}c_j=K$ 的充分必要条件是存在 $\{1,2,\cdots,n+2\}$ 的一个子集 S' 使得 $\sum_{j\in S'}c_j=\sum_{j\notin S'}c_j$. 下面证明满足定义 1.9 的(2).

充分性：由于 $5M - 2K > M = \sum_{j=1}^{n} c_j$，因此存在$\{1,2,\cdots,n\}$的一个子集 S，使得

$$\sum_{j\in S} c_j + c_{n+2} = \sum_{j\notin S, j\neq n+1,n+2} c_j + c_{n+1}. \tag{1.40}$$

所以，计算上式可得 $\sum_{j\in S} c_j = K$.

必要性：由(1.40)，令 $S' = S \cup \{n+2\}$，得结论. □

例 1.4 装箱问题的判定问题可以叙述为：给定 K 及 n 个物品，物品尺寸为 $0 < b_1, b_2, \cdots, b_n \leqslant 1$，是否能在 K 个尺寸为 1 的箱子中装入这 n 个物品？

例 1.33 装箱判定问题是 NPC.

装箱判定问题属于 NP 易证. 我们通过集合划分问题多项式转换装箱判定问题而证明结论. 由例 1.32 的证明，当集合划分问题的所有整数为正时，问题还是 NPC. 对集合划分问题的任一实例：n 个正整数 $c_1, c_2, \cdots c_n$，映射装箱问题的一个判定实例：n 个物品的尺寸满足

$$b_i = \frac{2c_i}{\sum_{j=1}^{n} c_j}, \quad i = 1,2,\cdots,n,$$

箱子尺寸为 1，是否能在两个箱子内装完所有物品？显然上面的映射满足定义 1.9 的(1). 现验证满足定义 1.9 的(2). 证明下列推断：存在$\{1,2,\cdots,n\}$的一个子集 S 使 $\sum_{j\in S} c_j = \sum_{j\notin S} c_j$ 的充分必要条件是可在两个尺寸为 1 的箱子里装入尺寸为

$$b_i = \frac{2c_i}{\sum_{j=1}^{n} c_j}, \quad i = 1,2,\cdots,n$$

的 n 个物品.

充分性：因为 $\sum_{j=1}^{n} b_j = 2$，则每个箱子都无空隙. 记第一个箱子

中的物品集为 S，有 $\sum_{j\in S} c_j = \sum_{j\notin S} c_j = 1$.

必要性：明显. □

例 1.5 的约束单机排序问题对应的判定问题是：给定 $d_i, i = 1,2,\cdots,n, c_t, t = 1,2,\cdots$ 和 T，在模型的约束条件下，能否在 T 个工作日内完成 n 个产品的加工？

例 1.34　约束单机排序的判定问题是 NPC.

约束单机排序的判定问题属于NP是很易验证的. 当 $c_t(t = 1, 2,\cdots)$ 全部相同时，便成为装箱问题的一种特殊情况，因此，装箱问题就多项式转换为约束单机排序问题的这一特殊情况. 所以，结论成立. □

由于 NPC 里包含很多著名的组合优化问题，经过几代数学家的努力，迄今没有找到多项式时间算法，人们猜想：NPC 中的任何一个问题没有多项式时间算法，即 $\mathrm{P} \cap \mathrm{NPC} = \varnothing$.

在讨论问题复杂性时，我们一直用判定问题去讨论，大量的组合最优化问题都有目标函数，这一点不同于判定问题. 上面的讨论都是将组合优化问题化成相应的判定问题，它们之间有什么关系呢？

线性规划问题(LP)：

$$\begin{aligned} &\min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b} \\ &\qquad \boldsymbol{x} \geqslant \boldsymbol{0} \end{aligned}$$

判定问题(DP)：

给定整数 z，是否有

$$\{\boldsymbol{x} \mid \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leqslant z, \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \boldsymbol{x} \geqslant \boldsymbol{0}\} \neq \varnothing ?$$

原问题 LP 同判定问题 DP 的联系是 z：如果原问题可以求解，则显而易见地回答了判定问题. 反过来如何？在文献[10]中，通过对线性规划基础可行解每一个分量界、目标值的估计，得到结论：LP 存在多项式时间最优算法的充分必要条件是 DP 存在多项式时间最优算法. 两者之间的联系是通过对目标值上下界的估计，在存在最优解时，用二分法给出 z，并通过二分法的逼近求出最优值.

用二分法迭代的次数是实例输入长度的多项式函数.下面给出严格的证明.

性质 1.2 假设整系数线性规划有两个基本可行解 x^1 和 x^2,并且对某一个整数 K 满足

$$K2^{-2L} < c^T x^1, c^T x^2 \leqslant (K+1)2^{-2L},$$

其中,$L = mn + m + n + \log_2|p|$,$p$ 为 A,b,c 中非零分量的乘积,则有 $c^T x^1 = c^T x^2$.

证明 反证若 $c^T x^1 \neq c^T x^2$,由性质 1.1,每一个基本可行解分母的绝对值不超过 2^L,则有 $|c^T x^1 - c^T x^2| \geqslant 2^{-2L}$,同性质 1.2 的假设矛盾. □

考虑区间$[-2^{4L}, 2^{4L}]$内的整数点,由性质 1.1 所有基本可行解对应的目标值落入区间$(-2^{2L}, 2^{2L})$内,此时只要 K 选区间$[-2^{4L}, 2^{4L}]$内的整数点,性质 1.2 的条件满足.结合性质 1.1、性质 1.2 和区间$[-2^{4L}, 2^{4L}]$条件,分下面三种情况讨论.

情况 1:$K = 2^{4L}, z = K2^{-2L}$ 时,DP 是一个"否"实例,则 LP 无可行解.

情况 2:$K = -2^{4L}, z = K2^{-2L}$ 时,DP 还是一个"是"实例,则 LP 无下界.

情况 3:以上两种情况都不满足时,此时可以用下面的二分算法:

二分算法

STEP1 $a = -2^{4L}, b = 2^{4L}; K = \dfrac{b-a}{2} = 0$;

STEP2 若 $b - a \leqslant 1$ 时停止,否则 $z = K2^{-2L}$ 时,若 DP 为"是"实例,则 $a := a, b := K$;

若 DP 为"否"实例时,则 $a := K, b := b, K = \dfrac{b-a}{2}$,重复 STEP2.

二分算法的迭代次数为 $\log_2 2^{4L+1} = 4L + 1$. 于是，有 LP 和 DP 复杂性的关系如下：

定理 1.1　LP 有多项式时间最优算法的充分必要条件是 DP 有多项式时间算法.

证明　必要性：当 LP 存在一个多项式时间最优算法时，则可以求解 LP 的最优解，于是由 LP 这个多项式时间算法加上例 1.22 的多项式时间验证算法就构成 DP 的多项式时间算法.

充分性：若 DP 存在一个多项式时间算法时，则由上面三种情况的讨论，得到 LP 也有多项式时间最优算法. □

正是基于这一点，建立组合优化问题同判定问题的关系

组合优化问题(OP)：　判定问题(RP)：

$\min f(x)$

s.t. $g(x) \geqslant 0$　　给定整数 L，是否有

$x \in D$　　$\{x \mid f(x) \leqslant L, g(x) \geqslant 0, x \in D\} \neq \varnothing$?

对于优化问题，若其判定问题是 NPC 或 NP-hard，则知优化问题的难度不低于判定问题，此时，称组合优化问题为 NP-hard.

算法复杂性的四类问题关系可用示意图 1.4 表示.

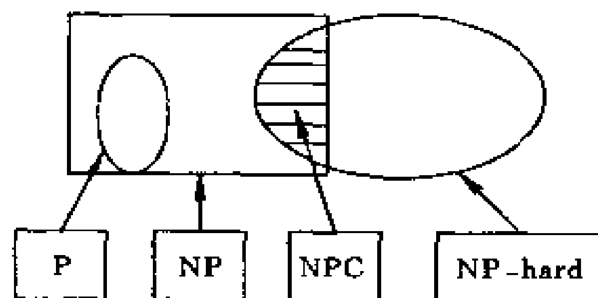


图 1.4　算法复杂性示意图

示意图 1.4 表示 P ⊂ NP，NPC ⊂ NP-hard，NP 与 NP-hard 的公共部分为 NPC. 在 NP 中，除 P 和 NPC，还有一部分问题的复杂性是未知的. 作为本节的结束，我们给出一个假设. 虽说这个假设

是组合优化问题中有待研究的一个热点，到目前为止，大家一般都接受这个假设！

假设 $P \neq NP$.

按 NPC 的定义容易证明：若 $P \cap NPC \neq \emptyset$，则 $NPC = P$. 有关算法及复杂性的理论的详细讨论，可以参考文献[5].

1.6 小 结

计算复杂性理论是组合最优化的基础. 只有了解所研究问题的复杂性才可能有针对性地设计算法，才能提高工作效率，起到事半功倍的作用. 一些实际应用人员往往忽略这一基础工作，这不仅仅是因为数学的难度，更重要的是一种观念在起主导作用. 这种观念是：无论解的效果如何，能找一个算法或比较已有的各种方法求得一个解即可. 由这一章的讨论，这对 NP-hard 问题是不得已的办法. 应该注意的是有些问题是多项式时间可解，而此时再采用这样的方法势必造成解的效果较差和计算时间的浪费. 一个多项式时间可解的问题一般不必用那些求解 NP-hard 问题的启发式算法.

本书的后续章节将介绍目前新兴的全局优化启发式算法：禁忌搜索、模拟退火、遗传算法、人工神经网络和用于估计下界的拉格朗日松弛算法. 它们主要用于求解 NP-hard 组合优化问题. 对于启发式算法的评价、最坏情况分析适用于比较简单的组合优化问题，它需要较深厚的数学基础. 另一种理论分析方法是概率分析方法，同样因为较深的数学理论和研究的难度，使得这一方法难以推广应用. 数值计算评价的方法简单、易行、易于理解和有较好的说服力，因此得到研究者和实际应用者的广泛使用.

练　习　题

1. 举例说明什么是组合最优化问题？并以此说明问题与实例的区别.

2. 给出求解下列每一个问题的算法. 在每一个问题里固定输入的表示，即给出它的一个例子，并将算法所需时间的上界表示为输入长度的函数.

(1) 给定平面上 $n(>2)$ 条直线 $\{a_i x + b_i y = c_i \mid i = 1,2,\cdots,n\}$，其中 $\{a_i, b_i, c_i \mid i = 1,2,\cdots,n\}$ 为整数. 这些线有公共点吗？

(2) 给定一个整数 p，p 是素数吗？

(3) 给定 n 个整数，选出最大数.

(4) 例 1.28 背包问题贪婪算法.

3. 给出集合 $P = \{x \in \mathbf{R}_+^n \mid \sum_{j=1}^{n} a_j x_j \leqslant b\}$ 系数输入长度及极点输入长度同系数输入长度关系的估计，其中 $a_j (j = 1,2,\cdots,n)$，b 为正整数.

4. 证明：对一切给定的整数 $k \geqslant 2$ 和 $\epsilon > 0$，有 $(\log n)^k = O(n^\epsilon)$，$(\log k)^{\log n} = O(k^n)$.

5. f 和 g 的定义如下

$$f(n) = \begin{cases} n^2, & \text{若 } n \text{ 为素数}, \\ n^3, & \text{其余}, \end{cases} \qquad g(n) = \begin{cases} n^2, & \text{若 } n \text{ 为奇数}, \\ n^3, & \text{其余}. \end{cases}$$

下述哪个结论对？

(1) $f(n) = O(g(n))$，

(2) $g(n) = O(f(n))$，

(3) $f(n), g(n) = O(n^4)$，

(4) $f(n), g(n) = O(n^2)$.

6. 判定问题“一个图中是否有从一个起点经过至少一个其他

节点以后再回到这个起点的一个圈?"属于P.给出一个多项式时间的算法并估计算法的复杂性.

7. 证明,表达式

$$(x_1+\overline{x_2})\cdot(\overline{x_1}+x_2)\cdot(x_1+x_2)\cdot(\overline{x_1}+\overline{x_2})$$

不是适定的.

8. 证明:若存在计算例1.8的TSP判定问题的多项式时间算法,则存在求TSP最优路径的多项式时间算法.

9. 给出一个由0 1背包问题(例1.1)到一般整数背包问题(例1.33)的多项式转换.

10. 证明集合覆盖问题属于NPC.集合覆盖问题为:给定一个$m\times n$的0-1矩阵A、整系数$(c_1,c_2,\cdots,c_n)$和一个整数K,是否存在取值为0-1的变量$\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$,使得$A\boldsymbol{x}\geqslant 1$且$\sum_{j=1}^{n}c_jx_j\leqslant K$?

11. 哈密顿(Hamilton)问题为:给定一个无向图$G=(N,E)$,其中$N=\{1,2,\cdots,n\}$为所有的结点组成的集合,$E=\{(i,j)|i,j\in N\}$为边集合,是否存在一个闭圈通过所有结点正好一次?假设哈密顿问题是NPC,证明:TSP属于NP-hard问题.

12. 证明:

$$\max\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}$$

$$\text{s.t.}\ \sum_{i=1}^{n}x_{ij}=1,\forall\ j,$$

$$\sum_{j=1}^{n}x_{ij}=1,\forall\ i,$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}t_{ij}x_{ij}\leqslant T,$$

$$x_{ij}\in\{0,1\}\forall\ i,j$$

是 NP-hard 问题.

13. 我们一再强调后续章节介绍的禁忌搜索、模拟退火、遗传算法和人工神经网络是全局优化算法.全局最优的含义是什么?计算复杂性怎样?求解组合优化问题的最优解所付出的代价是什么?

14. NP-hard 或 NPC 问题的每一个实例是否都不存在多项式时间的最优算法?

参考文献

1. Osman H I. Metaheuristics: a bibliography. Annals of Operations Research, 1996, 63:513～623
2. 邢文训. 能力约束单机排序问题研究. 清华大学博士论文, 1997
3. Klee V, Minty G J. How good is the simplex algorithm? In: Shisha O ed. Inequalities III. New York: Academic Press Inc, 1972:150～175
4. Khachian L G. A polynomial algorithm for linear programming. Doklady Akad. Nauk USSR, 1979, 244(5):1093～1096
5. 加里 M R,约翰逊 D S(张立昂等译). 计算机和难解性:NP-C 理论导论. 科学出版社,1987
6. Lin S, Kernighan B W. An effective heuristic algorithm for the traveling salesman problem. Operations Research, 1973, 21:498～516
7. Reeves C R(Ed). Modern Heuristic Techniques for Combinatorial Problems. Oxford: Blackwell Scientific Publications, 1993
8. Polya G. How to Solve It? Princeton: Princeton University Press,1948
9. Lawler E L. Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, 1976
10. Papadimitriou C H, Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. New Jersey: Prentice-Hall INC, 1982
11. Borgwardt K H. Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex method. Math Oper Res, 1982, 7:441～462

12. Solomon M M. Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res, 1987, 35:254~265
13. 索尔·加斯(王建华等译). 线性规划:方法及应用. 高等教育出版社, 1990
14. Cook S A. The complexity of theorem proving procedures. In: Proc 3rd ACM Symp on the Theory of Computing ACM. 1971:151~158

第2章

禁忌搜索算法

禁忌搜索(tabu search)算法是局部邻域搜索算法的推广，是人工智能在组合优化算法中的一个成功应用. Glover[1] 在 1986 年首次提出这一概念，进而形成一套完整算法，详见文献[2,3]. 禁忌搜索算法的特点是采用了禁忌技术. 所谓禁忌就是禁止重复前面的工作. 为了回避局部邻域搜索陷入局部最优的主要不足，禁忌搜索算法用一个禁忌表记录下已经到达过的局部最优点，在下一次搜索中，利用禁忌表中的信息不再或有选择地搜索这些点，以此来跳出局部最优点. 禁忌搜索算法是一种人工智能的算法，因此，有很多技术的细节问题有待下面讨论.

2.1 局部搜索

在这一章中，除特别强调外，我们都假设算法用以解决如下组合最优化问题：

$$\min f(x)$$
$$\text{s.t.}\ g(x) \geqslant 0,$$
$$x \in D.$$

其中 $f(x)$ 为目标函数，$g(x)$ 为约束方程，D 为定义域.

因为禁忌搜索算法中用到局部搜索算法，我们首先介绍局部搜索算法. 该算法可以简单的表示为：

局部搜索算法

STEP1 选定一个初始可行解 x^0；记录当前最优解 $x^{best}:=x^0$，令 $P=N(x^{best})$；

STEP2 当 $P=\varnothing$ 时，或满足其他停止运算准则时，输出计算结果，停止运算；否则，从 $N(x^{best})$ 中选一集合 S，得到 S 中的最优解 x^{now}；若 $f(x^{now})<f(x^{best})$，则 $x^{best}:=x^{now}$，$P:=N(x^{best})$；否则，$P:=P-S$；重复 STEP2.

在局部搜索算法中，STEP1 的初始可行解选择可以采用随机的方法，也可用一些经验的方法或是其他算法所得到的解. STEP2 中的集合 S 选取可以大到是 $N(x^{best})$ 本身，也可以小到只有一个元素，如用随机的方法在 $N(x^{best})$ 中选一点. 从直观可以看出，S 选取得小将使每一步的计算量减少，但可比较的范围很小；S 选取大时每一步计算时间增加，比较的范围自然增加. 这两种情况的应用效果依赖于实际问题. 在 STEP2 中，其他停止准则是除 STEP2 的 $P=\varnothing$ 以外的其他准则. 这些准则的给出往往取决于人们对算法的计算时间、计算结果的要求. 通过下面的例子来理解局部搜索算法.

例 2.1 5个城市的对称 TSP 数据如图 2.1.

对应的距离矩阵为

$$D=(d_{ij})=\begin{bmatrix}0 & 10 & 15 & 6 & 2\\ 10 & 0 & 8 & 13 & 9\\ 15 & 8 & 0 & 20 & 15\\ 6 & 13 & 20 & 0 & 5\\ 2 & 9 & 15 & 5 & 0\end{bmatrix}$$

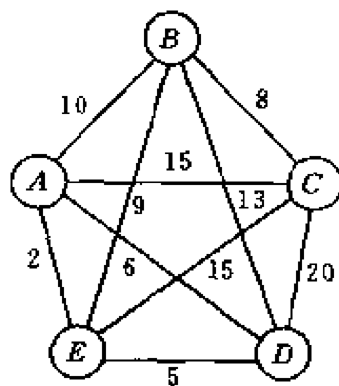


图 2.1 五城市 TSP

初始解为 $x^{best}=(ABCDE)$，$f(x^{best})=45$. 在本例中，邻域映

射定义为对换两个城市位置的 2-opt. 选定 A 城市为起点，我们用两种情况解释局部搜索算法.

情况 1　全邻域搜索，即 $S := N(\boldsymbol{x}^{best})$.

第一循环：$N(\boldsymbol{x}^{best}) = \{(ABCDE), (ACBDE), (ADCBE), (AECDB), (ABDCE), (ABEDC), (ABCED)\}$，对应目标函数值为：$f(\boldsymbol{x}) = \{45, 43, 45, 60, 60, 59, 44\}$.

$\boldsymbol{x}^{best} := \boldsymbol{x}^{now} = (ACBDE)$.

第二循环：$N(\boldsymbol{x}^{best}) = \{(ACBDE), (ABCDE), (ADBCE), (AEBDC), (ACDBE), (ACEDB), (ACBED)\}$，对应目标函数值为：$f(\boldsymbol{x}) = \{43, 45, 44, 59, 59, 58, 43\}$

$\boldsymbol{x}^{best} := \boldsymbol{x}^{now} = (ACBDE)$.

此时，$P = N(\boldsymbol{x}^{best}) - S$ 为空集，于是所得解为 $(ADCBE)$，目标值为 43.

情况 2　一步随机搜索.

$\boldsymbol{x}^{best} = (ABCDE), f(\boldsymbol{x}^{best}) = 45$

第一循环：由于采用 $N(\boldsymbol{x}^{best})$ 中的一步随机搜索，可以不再计算 $N(\boldsymbol{x}^{best})$ 中每一点的值. 若从中随机选一点，如 $\boldsymbol{x}^{now} = (ACBDE)$. 因 $f(\boldsymbol{x}^{now}) = 43 < 45$，所以 $\boldsymbol{x}^{best} := (ACBDE)$.

第二循环：若从 $N(\boldsymbol{x}^{best})$ 中又随机选一点 $\boldsymbol{x}^{now} = (ADBCE)$，$f(\boldsymbol{x}^{now}) = 44 > 43$. $P = N(\boldsymbol{x}^{best}) - \{\boldsymbol{x}^{now}\}$. 最后得到的解为 $(ACBDE)$.

□

局部搜索算法的优点是简单易行，容易理解，但其缺点是无法保证全局最优性.

例 2.2　四城市非对称 TSP 如图 2.2.

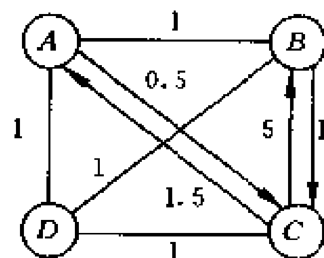


图 2.2　四城市 TSP

距离矩阵为

$$D=(d_{ij})=\begin{bmatrix}0 & 1 & 0.5 & 1\\ 1 & 0 & 1 & 1\\ 1.5 & 5 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}.$$

若初始解为 $x^{best}=(ABCD)$ 并且假设城市 A 为起始点，$f(x^{best})=4$.

邻域 $N(x^{best})=\{(ABCD),(ACBD),(ADCB),(ABDC)\}$ 中，局部最优解是$(ABCD)$. 读者可以按例 2.1 局部搜索讨论的两种情况进行验证，该算法终止时的解是局部最优解$(ABCD)$. 而全局最优解是 $x^{best}=(ACDB)$，$f(x^{best})=3.5$. □

改变局部搜索中只按下降规则转移状态的一个方法是蒙特卡罗(Monte Carlo)方法，主要变化是局部搜索算法的第二步为：

蒙特卡罗算法

STEP2 当满足停止运算准则时，停止运算；否则，从 $N(x^{best})$ 中随机选一点 x^{now}；若 $f(x^{now})\leqslant f(x^{best})$，则 $x^{best}:=x^{now}$；否则，根据 $f(x^{now})-f(x^{best})$ 以一定的概率接受 $x^{now}(x^{best}:=x^{now})$；返回 STEP2.

蒙特卡罗算法是以一定的概率接受一个较坏的状态，如以均匀的概率

$$p=\frac{1}{|N(x^{best})|}$$

从 $N(x^{best})$ 中选任意一点，以概率

$$p=\min\left\{1,\exp\left\{-\frac{f(x^{now})-f(x^{best})}{t}\right\}\right\}$$

接受 x^{now}. 模拟退火算法就是采用这样的搜索方法，有关的性质将

在第 3 章中讨论. 这样可能遍历所有的状态. 不同于蒙特卡罗随机搜索算法的思想,禁忌搜索则是用确定性的方法跳出局部最优解.

2.2　禁忌搜索

禁忌搜索是一种人工智能算法,是局部搜索算法的扩展. 它的一个重要思想是标记已得到的局部最优解,并在进一步的迭代中避开这些局部最优解. 如何避开和记忆这些点是本章主要讨论的问题. 首先,用一个示例来理解禁忌搜索算法.

例 2.3　(例 2.2 续) 假设: 初始解 $\boldsymbol{x}^0=(ABCD)$,邻域映射为两个城市位置对换,始终点都为 A 城市. 目标值为 $f(\boldsymbol{x}^0)=4$. 城市间的距离为:

$$\boldsymbol{D}=(d_{ij})=\begin{bmatrix}0 & 1 & 0.5 & 1\\ 1 & 0 & 1 & 1\\ 1.5 & 5 & 0 & 1\\ 1 & 1 & 1 & 0\end{bmatrix}.$$

第 1 步:

解的形式

A	B	C	D

$f(\boldsymbol{x}^0)=4$

禁忌对象及长度

	B	C	D
A			
B			
C			

候选集

对换	评价值
C,D	4.5★
B,C	7.5
B,D	8

此处评价值为目标值. 由于假设了 A 城市为起终点,故候选集中最多有两两城市对换对 3 个. 分别对换城市顺序并按目标值由小到大排列,三个评价值都劣于原值,在原有的局部搜索算法中,此时已达到局部最优解而停止. 但现在,我们允许从候选集中选一

个最好的对换——CD城市的位置交换，用★标记入选的对换．此时，解从$(ABCD)$变化为$(ABDC)$，目标值上升，但此法可能跳出局部最优．

第 2 步：

解的形式

A	B	D	C

$f(x^1)=4.5$

禁忌对象及长度

	B	C	D
A			
B			
C			3

候选集

对换	评价值
B,C	3.5★
B,D	4.5
C,D	4.5T

由于第 1 步中选择了CD交换，于是，我们希望这样的交换在下面的若干次迭代中不再出现，以避免计算中的循环，CD成为禁忌对象并限定在 3 次迭代计算中不允许CD或DC对换．在对应位置记录 3．在$N(x^1)$中又出现被禁忌的CD对换，故用 T 标记而不选此交换．在选择最佳的候选对换后，到第 3 步．

第 3 步：

解的形式

A	C	D	B

$f(x^2)=3.5$

禁忌对象及长度

	B	C	D
A			
B		3	
C			2

候选集

对换	评价值
B,C	4.5T
B,D	7.5★
C,D	8T

新选的BC对换被禁后，CD在被禁一次后还有二次禁忌．虽说候选集中的评价值都变坏，但为达到全局最优，还是从中选取．由于BC和CD对换被禁，只有BD对换入选．

第 4 步：

解的形式

A	C	B	D

$f(\boldsymbol{x}^3)=7.5$

禁忌对象及长度

	B	C	D
A			
B		2	3
C			1

候选集

对换	评价值
B,D	3.5T
B,C	4.5T
C,D	4.5T

此时，所有候选对换被禁，怎么办？ □

通过这一个示例，我们会产生如下问题：

· 本例禁忌对象是对换，如 BC 对换造成 $ABCD$ 到 $ACBD$ 的变化，禁忌 BC 对换包括如下城市间顺序的对换：$ACBD$ 到 $ABCD$、$ABCD$ 到 $ACBD$、$ADBC$ 到 $ADCB$、$ADCB$ 到 $ADBC$、$ABDC$ 到 $ACDB$、$ACDB$ 到 $ABDC$ 等的变化. 若 $ACBD$ 是刚才由 $ABCD$ 变化而来的，结合解的变化，禁忌 $ACBD$ 到 $ABCD$ 和 $ABCD$ 到 $ACDB$ 的变化是可以接受的. 但对其他变化的禁忌是否会影响求解的效率？如 $ADBC$ 到 $ADCB$ 是否允许？

这一个问题说明禁忌对象是禁忌算法中一个基本的因素.

· 禁忌的次数如何选取？若在例 2.3 中将禁忌次数从 3 更改为 2，则有下列情形：

例 2.4

第 4 步：

解的形式

A	C	B	D

$f(\boldsymbol{x}^3)=7.5$

禁忌对象及长度

	B	C	D
A			
B		1	2
C			0

候选集

对换	评价值
B,D	3.5T
B,C	4.5T
C,D	4.5★

第 5 步：

解的形式

A	D	B	C

$f(x^4)=4.5$

禁忌对象及长度

	B	C	D
A			
B		0	1
C			2

候选集

对换	评价值
B,D	4.5T
C,D	7.5T
B,C	8★

再迭代一步，又回到状态（$ABCD$），此时出现循环. □

由例 2.3 和例 2.4 的计算可以看出，禁忌搜索算法是局部搜索算法的变形. 应该注意到禁忌搜索算法计算中的关键点：禁忌对象、长度和候选集成为算法的主要特征. 围绕这些特征需要考虑下列因素.

（1）是否有其他形式的候选集？上面的例子是将所有可对换的城市对作为候选集，再从候选集中没有被禁的对换对中选最佳. 对 n 个城市的 TSP，这样构造候选集使得每个集中有 C_{n-1}^2 个交换对. 为了节省每一步的计算时间，有可能只在邻域中随机选一些对换，而不一定是比较邻域中的所有对换.

（2）禁忌的长度如何确定？如果在算法中记忆下搜索到的当前最优解，极端的两种情况是：一是将所有的对换个数作为禁忌长度，此时等价于将候选集中的所有的对换遍历；另外则取为 1，这等价于局部搜索算法.

（3）是否有评价值的其他替代形式？上面例中用目标值作为评价值. 有时计算目标值的工作量较大，或无法接受计算目标值所花费的时间，于是需要其他的方法.

（4）被禁的对换能否再一次解禁？如在例 2.3 的第 4 步中，候选集中的交换都被禁忌，若此时停止，得到的解甚至不是一个局部最优解. 候选集中 BD 对换的评价值最小，是否不考虑对 BD 的禁

忌而选择这个对换?有这样的直观现象.当搜索到一个局部最优解后,它邻域中的其他状态都被禁,我们是否解禁一些状态以便跳出局部最优?解禁的功能就是为了获得更大的搜索范围,以免陷入局部最优.

(5) 如何利用更多的信息?在禁忌搜索算法中,还可记录其他一些信息.如一个被禁对象(交换)被禁的次数,评价值变化的大小等.如果在例 2.3 中记忆同一个对换出现的此数,我们可以得到如下的一个有关禁忌对象的信息,

	A	B	C	D
A	×			
B		×	2	3
C		5	×	1
D		2	3	×

其中,矩阵右上角记录禁忌的长度,矩阵的左下角部分出现的数据表示对换被选为最佳的次数.B 与 C 对应 5 表示 BC 交换 5 次成为最佳候选.这些数字提供了状态出现的频率,反映解的一些性质.

(6) 终止原则,即一个算法停止的条件,怎样给出?

综合上面的讨论,禁忌算法的特征由禁忌对象和长度、候选集和评价函数、停止规则和一些计算信息组成.禁忌表特别指禁忌对象及其被禁的长度.禁忌对象是指变化的状态,如上面例子中的两个城市的对换.候选集中的元素依评价函数而确定,根据评价函数的优劣选择一个可能替代被禁对象的元素,是否替代取决于禁忌的规则和其他一些特殊规则,在后续部分将介绍一个特殊规则——特赦原则.计算中的一些信息,如被禁对象对应的评价值、被禁的频率等,将对禁忌的长度和停止规则提供帮助.

禁忌搜索算法

STEP1 选定一个初始解 $\boldsymbol{x}^{now}$ 及给以禁忌表 $H=\varnothing$；

STEP2 若满足停止规则，停止计算；否则，在 $\boldsymbol{x}^{now}$ 的邻域 $N(H,\boldsymbol{x}^{now})$ 中选出满足禁忌要求的候选集 Can_ $N(\boldsymbol{x}^{now})$；在 Can_ $N(\boldsymbol{x}^{now})$ 中选一个评价值最佳的解 $\boldsymbol{x}^{next}$，$\boldsymbol{x}^{now}:=\boldsymbol{x}^{next}$；更新历史记录 H，重复 STEP 2.

禁忌算法的STEP2中，$\boldsymbol{x}^{now}$ 的邻域 $N(H,\boldsymbol{x}^{now})$ 中满足禁忌要求的元素包含两类：一类是那些没有被禁忌的元素，另一类是可以被解除禁忌的元素. 详细的技术问题将在第2.3节讨论.

比较局部搜索算法、蒙特卡罗算法和禁忌搜索算法，它们的主要区别是第二步对接受点选择的原则不同. 为了给出禁忌搜索算法的全局最优定理，先介绍连通的概念.

定义 2.1 集合 C 称为相对邻域映射 $N: \boldsymbol{x}\in C\rightarrow 2^{C}$ 是连通的，若对 C 中的任意两点 $\boldsymbol{x},\boldsymbol{y}$，存在 $\boldsymbol{x}=\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_l=\boldsymbol{y}$，使得 $N(\boldsymbol{x}_i)\cap N(\boldsymbol{x}_{i+1})\neq\varnothing, i=1,2,\cdots,l-1$.

定理 2.1 （最优定理）在禁忌搜索算法中，若可行解区域相对 Can_ $N(\boldsymbol{x}^{now})$ 是连通的，且 H 的记录充分大，则一定可以达到全局最优解.

证明非常直观. 虽说从理论上保证全局最优，但若使得 H 的记录充分大，也就是遍历所有的状态，这不是我们希望的. 我们的期望是用更少的花费得到我们期望的解.

2.3 技术问题

禁忌搜索算法是一种人工智能算法，因此，实现的技术问题是算法的关键. 本节按禁忌对象、候选集合的构成、评价函数的构造、

特赦规则、记忆频率信息和终止规则等分别给予介绍和讨论.

2.3.1　禁忌对象、长度与候选集

禁忌表中的两个主要指标是禁忌对象和禁忌长度. 顾名思义，禁忌对象指的是禁忌表中被禁的那些变化元素. 因此首先需要了解状态是怎样变化的. 我们将状态的变化分为解的简单变化、解向量分量的变化和目标值变化三种情况. 在这三种变化的基础上，讨论禁忌对象. 本小节同时介绍禁忌长度和候选集确定的经验方法.

1. 解的简单变化

这种变化最为简单. 假设 $\boldsymbol{x},\boldsymbol{y}\in D$，其中 D 为优化问题的定义域，则简单解变化为

$$\boldsymbol{x}\rightarrow\boldsymbol{y}$$

是从一个解变化到另一个解. 这种变化在局部搜索算法中经常采用，如例 2.1 第一循环中从 $(ABCDE)$ 变化到 $(ACBDE)$. 这种变化将问题的解看成变化最基本因素.

2. 向量分量的变化

这种变化考虑的更为精细，以解向量的每一个分量为变化的最基本因素. 仅以 $(ABCDE)$ 变化到 $(ACBDE)$ 为例，它的变化实际是由 B 和 C 的对换引起. 但 B 和 C 对换可以引起更多解的简单变化，如

$$(ABCDE)\rightarrow(ACBDE),$$
$$(ABDCE)\rightarrow(ACDBE),$$
$$(ACBED)\rightarrow(ABCED),$$

等. 设原有的解向量为 $(x_1,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)$，用数学表达式来描述向量分量的最基本变化为：

$$(x_1,\cdots,x_{i-1},x_i,x_{i+1},\cdots,x_n)\rightarrow(x_1,\cdots,x_{i-1},y_i,x_{i+1},\cdots,x_n),$$

即只有第 i 个分量发生变化. 向量的分量变化包含多个分量发生变化的情形.

部分优化问题的解可以用一个向量形式 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^T\in\{0,1\}^n$ 表示.解与解之间的变化可以表示某些分量的变化,如用分量从 $x_j=0$ 变化为 $x_j=1$ 或从 $x_k=1$ 变化为 $x_k=0$,或是两者的结合.可以通过下面两个例子理解.

例 2.5 例 1.1 的 0-1 背包问题的状态变化是向量分量变化形式.如果每次只允许一个变量变化,即,$N:\boldsymbol{x}\in D\to N(\boldsymbol{x})=\{\boldsymbol{y}\mid\sum_{i=1}^{n}|y_i-x_i|\leqslant 1\}\in 2^D$,则变化的变量只可能由 $x_j=0$ 变化为 $y_j=1$ 或由 $x_j=1$ 变化为 $y_j=0$.此时,状态变化的情况是一个变量 $x_j=0$ 变化为 $x_j=1$ 或一个变量 $x_k=1$ 变化为 $x_k=0$. □

例 2.6 TSP 问题采用例 1.28 的 2-opt 位置交换规则.以例 2.2 的四城市 TSP 数据为例.当一个解为 $(ABCD)$ 时,两个城市 BD 的交换可以理解为:用一个 $\{0,1\}^{n\times(n-1)}$ 的向量表示解,则 $(x_{AB}=1,x_{BC}=1,x_{CD}=1,x_{DA}=1)$,其他分量为零;城市 BD 的交换为 $(x_{AD}=1,x_{DC}=1,x_{CB}=1,x_{BA}=1)$,其他分量为零;分量的变化是:由 $x_{AB}=1$ 变化为 $x_{AB}=0$,$x_{BC}=1$ 变化为 $x_{BC}=0$,$x_{CD}=1$ 变化为 $x_{CD}=0$,$x_{DA}=1$ 变化为 $x_{DA}=0$,$x_{AD}=0$ 变化为 $x_{AD}=1$,$x_{DC}=0$ 变化为 $x_{DC}=1$,$x_{CB}=0$ 变化为 $x_{CB}=1$,$x_{BA}=0$ 变化为 $x_{BA}=1$.于是,一个 2-opt 交换共需 8 个分量发生变化.这种情况归类于多个分量变化的结合. □

3. 目标值变化

在优化问题的求解过程中,我们非常关心目标值是否发生变化,是否接近最优目标值.这就产生一种观察状态变化的方式:观察目标值或评价值的变化.就犹如等位线的道理一样,把处在同一等位线的解视为相同.这种变化是考察

$$H(a)=\{\boldsymbol{x}\in D\mid f(\boldsymbol{x})=a\},$$

其中,$f(\boldsymbol{x})$ 为目标函数.它的表面是两个目标值的变化,即从 $a\to b$,但隐含着两个解集合的各种变化 $\forall\ \boldsymbol{x}\in H(a)\to\forall\ \boldsymbol{y}\in H(b)$ 的

可能.

例 2.7 考虑目标函数 $f(x)=x^2$ 的目标值从 1 变化到 4，这里隐含着解空间中四个变化的可能

$$-1\to-2,1\to-2,-1\to2,1\to2.$$ □

以上三种状态变化的情形，第一种的变化比较单一，而第二和第三种变化则隐含着多个解变化的可能. 因此在选择禁忌对象时，可以根据实际问题采用适当的变化.

4. 禁忌对象的选取

由上面关于状态变化三种形式的讨论，禁忌的对象就可以是上面的任何一种. 现用示例来分别理解. 第一种情况考虑解为简单变化. 当解从 $x\to y$ 时，y 可能是局部最优解，为了避开局部最优解，禁忌 y 这一个解再度出现. 禁忌的规则是：当 y 的邻域中有比它更优的点时，则选择更优的解；当 y 为 $N(y)$ 的局部最优时，不再选 y，而选择比 y 较差的解. 见例 2.8.

例 2.8 （例 2.1 续）五城市 TSP 的距离矩阵为

$$\boldsymbol{D}=(d_{ij})=\begin{bmatrix}0&10&15&6&2\\10&0&8&13&9\\15&8&0&20&15\\6&13&20&0&5\\2&9&15&5&0\end{bmatrix},$$

禁忌对象为简单的解变化. 禁忌表 H 只记忆三个被禁的解，即禁忌长度为 3. 从 2-opt 邻域 $N(H,\boldsymbol{x}^{\text{now}})$ 中选出最佳的五个解组成候选集 $\text{Can_}N(\boldsymbol{x}^{\text{now}})$；初始解 $\boldsymbol{x}^{\text{now}}=\boldsymbol{x}^0=(ABCDE),f(\boldsymbol{x}^0)=45$.

第 1 步

$\boldsymbol{x}^{\text{now}}=(ABCDE),f(\boldsymbol{x}^{\text{now}})=45,H=\varnothing,\text{Can_}N(\boldsymbol{x}^{\text{now}})=\{(ABCDE;45),(ACBDE;43),(ADCBE;45),(ABEDC;59),(ABCED;44)\},\boldsymbol{x}^{\text{next}}=(ACBDE)$.

由于 H 为空集，从候选集中选最好的一个.

第 2 步

$x^{now}=(ACBDE)$，$f(x^{now})=43$，$H=\{(ACBDE;43)\}$，

Can_ $N(x^{now})=\{(ACBDE;43),(ACBED;43),(ADBCE;44),(ABCDE;45),(ACEDB,58)\}$，

由于$(ACBDE)$受禁，所以选 $x^{next}=(ACBED)$.

第 3 步

$x^{now}=(ACBED)$，$f(x^{now})=43$，$H=\{(ACBDE;43),(ACBED;43)\}$，

Can_ $N(x^{now})=\{(ACBED;43),(ACBDE;43),(ABCED;44),(AEBCD,45),(ADBEC;58)\}$.

由于 $H=\{(ACBDE;43),(ACBED;43)\}$ 受禁，所以选 $x^{next}=(ABCED)$.

第 4 步

$x^{now}=(ABCED)$，$f(x^{now})=44$，

$H=\{(ACBDE;43),(ACBED;43),(ABCED;44)\}$，

Can_ $N(x^{now})=\{(ACBED;43),(AECBD,44),(ABCDE,45),(ABCED;44),(ABDEC;58)\}$.

$x^{next}=(AECBD)$. 此时 H 已达 3 个解，新选入的解要替代出最早被禁的解.

第 5 步

$x^{now}=(AECBD)$，$f(x^{now})=44$，

$H=\{(ACBED;43),(ABCED;44),(AECBD;44)\}$，

Can_ $N(x^{now})=\{(AEDBC;43),(ABCED;44),(AECBD,44),(AECDB;44),(AEBCD;45)\}$.

$x^{next}=(AEDBC)$. □

如果禁忌第 2 种变化，则观察下例.

例 2.9 （例 2.8 续）数据同例 2.8. H 只记忆三对对换时，从

2-opt 邻域 $N(H,\boldsymbol{x}^{now})-\{\boldsymbol{x}^{now}\}$ 中选出最佳的五个状态对应的交换对组成候选集 Can_ $N(\boldsymbol{x}^{now})$，这一点不同于例 2.8，因为受禁的是交换对，因此不考虑没有变化的 $\boldsymbol{x}^{now}$. 初始解 $\boldsymbol{x}^{now}=\boldsymbol{x}^0=(ABCDE)$，$f(\boldsymbol{x}^0)=45$.

第 1 步

$\boldsymbol{x}^{now}=(ABCDE)$，$f(\boldsymbol{x}^{now})=45$，$H=\varnothing$，Can_ $N(\boldsymbol{x}^{now})=\{(ACBDE;43),(ADCBE;45),(AECDB;60),(ABEDC;59),(ABCED;44)\}$，$\boldsymbol{x}^{next}=(ACBDE)$.

由于 H 为空集，从候选集中选最好的一个，它是 B 与 C 的对换构成.

第 2 步

$\boldsymbol{x}^{now}=(ACBDE)$，$f(\boldsymbol{x}^{now})=43$，$H=\{(B,C)\}$，

Can_ $N(\boldsymbol{x}^{now})=\{(ACBED;43),(ADBCE;44),(ABCDE;45),(ACEDB,58),(AEBDC;59)\}$，

选 $\boldsymbol{x}^{next}=(ACBED)$. 它是 D 和 E 对换.

第 3 步

$\boldsymbol{x}^{now}=(ACBED)$，$f(\boldsymbol{x}^{now})=43$，$H=\{(B,C),(D,E)\}$，

Can_ $N(\boldsymbol{x}^{now})=\{(ACBDE;43),(ABCED;44),(AEBCD,45),(ADBEC;58),(ACEBD;58)\}$.

由于 $H=\{(B,C),(D,E)\}$ 受禁，所以选 $\boldsymbol{x}^{next}=(AEBCD)$.

前两步计算同例 2.8 的前两步相同，但第 3 步同例 2.8 的第 3 步不同. 因为根据禁忌表中的条件，禁忌选取由 $(ACBED)$ 经 BC 对换后的解 $(ABCED)$. 由此看出禁忌对换的范围要大于例 2.8 只对简单解禁忌的范围. □

例 2.9 的邻域定义和禁忌决定了：当一对元素 x 和 y 被禁忌后，包含两个元素的两种对换 x 与 y 交换和 y 与 x 交换，如禁忌 B 和 C 对换后，禁忌 C 与 B 对换和 B 与 C 对换. 禁忌 C 与 B 对换的出发点是：上一步已经对换的两个元素不能再对换回去，以免还原

到原有的解. 还以例 2.9 为例，假设在例 2.9 问题中，某一步得到解为$(ABCDE)$，迭代一步得到 $\boldsymbol{x}^{now}=(ACBDE)$ 时，此时，应该考虑禁忌 C 与 B 的对换，否则，可能回到$(ABCDE)$. 禁忌 B 与 C 对换的出发点是：上一步已经对 B 与 C 的对换进行了分析和评价，希望搜索有较大的遍历性，因此我们不再考虑 B 与 C 的对换，即禁忌这个对换.

在有些情况下，更细地将一对元素 x 和 y 的对换分成 x 与 y 对换和 y 与 x 对换两种情形，即考虑对换的方向. 因此，我们可以考虑只禁忌一个方向的对换，如 x 与 y 的对换或 y 与 x 的对换.

例 2.10 (例 2.9 续) 以第三种目标值的变化情形来继续观察例 2.9. H 只记忆三组元素(解及其目标值)，从 2-opt 邻域 $N(H,\boldsymbol{x}^{now})$ 中选出最佳的五个元素为候选集 Can_ $N(\boldsymbol{x}^{now})$；在 Can_ $N(\boldsymbol{x}^{now})$ 中选一个目标值最佳的解 $\boldsymbol{x}^{next}$. 初始解 $\boldsymbol{x}^{now}=\boldsymbol{x}^{0}=(ABCDE)f(\boldsymbol{x}^{0})=45$.

第 1 步

$\boldsymbol{x}^{now}=(ABCDE),f(\boldsymbol{x}^{now})=45,H=\varnothing$,

Can_ $N(\boldsymbol{x}^{now})=\{(ABCDE;45),(ACBDE;43),(ADCBE;45),(ABEDC;59),(ABCED;44)\}$,

$\boldsymbol{x}^{next}=(ACBDE)$.

由于 H 为空集，从候选集中选最好的一个.

第 2 步

$\boldsymbol{x}^{now}=(ACBDE),f(\boldsymbol{x}^{now})=43,H=\{43\}$,

Can_ $N(\boldsymbol{x}^{now})=\{(ACBDE;43),(ACBED;43),(ADBCE;44),(ABCDE;45),(ACEDB,58)\}$,

$\boldsymbol{x}^{next}=(ADBCE)$.

由于函数值 43 受禁，选候选集中不受禁的最佳函数值 44 的状态.

在此例中，采用禁忌目标值的方法所禁忌的范围比例 2.8 和

例 2.9 的受禁范围更大，这在这一例中明显地体现. □

所谓禁忌就是禁止重复前面达到局部最优的状态. 由于计算过程中解的状态在不断地变化，因此我们对造成状态的变化的对象进行禁忌. 上面已经讨论，状态变化的主要因素归结三种形式：简单的解的变化，解向量的分量变化和目标值的变化.

针对三种不同的状态变化方式，例 2.8、例 2.9 和例 2.10 体现了禁忌搜索算法在计算中的不同. 实际应用中，应根据具体问题采用一种方法. 从上面示例的计算中也可以看出，解的简单变化比解的分量变化和目标值变化的受禁忌范围要小，这可能造成计算时间的增加. 但它也给予了较大的搜索范围. 解分量的变化和目标值变化的禁忌范围要大，这减少了计算的时间，可能引发的问题是禁忌的范围太大以至陷在局部最优点.

由此可以得知，禁忌搜索算法中的技术很强. 因为 NP-hard 问题不可能奢望计算得到最优解，在算法的构造和计算的过程中，一方面要求尽量少的占用机器内存，这就要求禁忌长度、候选集合尽量小. 正好相反，禁忌长度过短造成搜索的循环，候选集合过小造成过早地陷入局部最优.

5. 禁忌长度的确定

禁忌长度是被禁对象不允许选取的迭代次数. 一般是给被禁对象 x 一个数(禁忌长度)t，要求对象 x 在 t 步迭代内被禁，在禁忌表中采用 $\mathrm{tabu}(x)=t$ 记忆，每迭代一步，该项指标做运算 $\mathrm{tabu}(x)=t-1$，直到 $\mathrm{tabu}(x)=0$ 时解禁. 于是，我们可将所有元素分成两类，被禁元素和自由元素. 有关禁忌长度 t 的选取，可以归纳为下面几种情况：

(1) t 为常数，如 $t=10$, $t=\sqrt{n}$，其中 n 为邻居的个数. 这种规则容易在算法中实现.

(2) $t\in[t_{\min},t_{\max}]$. 此时 t 是可以变化的数，它的变化是依据被禁对象的目标值和邻域的结构. 此时 $t_{\min}$, $t_{\max}$ 是确定的. 确定 $t_{\min}$,

t_{max} 的常用方法是根据问题的规模 T，限定变化区间 $[\alpha\sqrt{T}, \beta\sqrt{T}]$ $(0<\alpha<\beta)$. 也可以用邻域中邻居的个数 n 确定变化区间 $[\alpha\sqrt{n}, \beta\sqrt{n}]$ $(0<\alpha<\beta)$. 当给定了变化区间，确定 t 的大小主要依据实际问题、实验和设计者的经验. 如从直观可见，当函数值下降较大时，可能谷较深，欲跳出局部最优，希望被禁的长度较大.

(3) t_{min}，t_{max} 的动态选取. 有的情况下，用 t_{min}，t_{max} 的变化能达到更好的解. 它的基本思想同(2)类似.

禁忌长度的选取同实际问题、实验和设计者的经验有紧密的联系，同时它决定了计算的复杂性. 过短会造成循环的出现. 如 $f(1)=1$，$f(2)=3.5$，$f(3)=2.5$，$f(4)=2$，$f(5)=3$，$f(6)=6$，极端情况禁忌长度是 1，邻域为相距不超过 1 的整数点，一旦陷入局部最优点 $x=4$，则出现循环而无法跳出局部最优. 过长又造成计算时间较长. 上面(2)中给出的区间估计参数都是一些经验的估计.

6. 候选集合的确定

候选集合由邻域中的邻居组成. 常规的方法是从邻域中选择若干个目标值或评价值最佳的邻居入选. 如上面的 TSP 中采用 2-opt 邻域定义，一个状态的邻域中共有 C_n^2 个邻居，计算目标值后，例 2.8、例 2.9 和例 2.10 选择五个目标值最佳的邻居入选.

有时认为上面的计算量还是太大，则不在邻域的所有邻居中选择，而是在邻域中的一部分邻居中选择若干个目标值或评价值最佳的状态入选. 也可以用随机选取的方法实现部分邻居的选取.

2.3.2 评价函数

评价函数是候选集合元素选取的一个评价公式，候选集合的元素通过评价函数值来选取. 以目标函数作为评价函数是比较容易理解的. 目标值是一个非常直观的指标，但有时为了方便或易于

理解，会采用其他函数来取代目标函数.我们将评价函数分为基于目标函数和其他方法两类.

1. 基于目标函数的评价函数

这一类主要包含以目标函数的运算所得到的评价方法.如记评价函数为 $p(x)$，目标函数为 $f(x)$，则评价函数可以采用目标函数

$$p(x)=f(x);$$

目标函数值与 x^{now} 目标值的差值

$$p(x)=f(x)-f(x^{now}),$$

其中 x^{now} 是上一次迭代计算的解；目标函数值与当前最优解 x^{best} 目标值的差值

$$p(x)=f(x)-f(x^{best}),$$

其中 x^{best} 是目前计算中的最好解.

基于目标函数的评价函数的形成主要通过对目标函数进行简单的运算，它的变形很多.

2. 其他方法

有时计算目标值比较复杂或耗时较多，解决这一问题的方法之一是采用替代的评价函数.替代的评价函数还应该反映原目标函数的一些特性，如原目标函数对应的最优点还应该是替代函数的最优点.构造替代函数的目标是减少计算的复杂性.具体问题的替代函数构造依问题而定.它的一个例子是在生产计划中的约束批量计划与调度(capacitated lotsize planning and scheduling)模型中的应用.

例 2.11 简单的生产计划批量问题.一个工厂安排 n 个产品在 T 个计划时段里加工，其各个产品的需求量、加工费用和加工能力等之间的关系用下面数学模型(PP)描述，

$$\min\sum_{i=1}^{n}\sum_{t=1}^{T}(p_i x_{it}+s_t y_{it}+h_i I_{it}) \tag{2.1}$$

$$\text{s.t.}\ I_{it-1} + x_{it} - I_{it} = d_{it},\quad i = 1,2,\cdots,n,\ t = 1,2,\cdots,T, \tag{2.2}$$

$$\sum_{i=1}^{n} a_i x_{it} \leqslant c_t,\ t = 1,2,\cdots,T, \tag{2.3}$$

$$y_{it} = \begin{cases} 1, 若\ x_{it} > 0, \\ 0, 其它. \end{cases} \tag{2.4}$$

$$x_{it}, I_{it} \geqslant 0,\quad i = 1,2,\cdots,n,\ t = 1,2,\cdots,T.$$

其中，s_i 表示生产 i 产品需耗的生产准备费用；

h_i 表示单位 i 产品需耗的库存费用；

p_i 表示生产单位 i 产品需耗的费用；

x_{it} 表示第 t 时段，i 产品的生产批量；

I_{it} 表示第 t 时段，i 产品的库存量；

d_{it} 表示第 t 时段，i 产品的外部需求量；

a_i 表示生产单位 i 产品需耗的资源量；

c_t 表示第 t 时段能力的提供量.

上面模型中，x_{it}，I_{it} 为决策变量，所有参数为非负值. 目标(2.1)要求生产、生产准备和库存费用三项费用和最小，(2.2)为外部需求、生产和库存之间的平衡关系，(2.3)为资源约束，要求每个时段生产占用的能力不超过可提供的能力. PP 模型是一个混合整数规划问题. 因为 y_{it} 取 0，1 两个值，当 $Y^0 = (y_{it})$ 的值选定后，PP 模型为一个线性规划问题，对应的最优目标值为 $Z(Y^0)$. 这样，禁忌搜索算法可以只将(y_{it})看成决策变量进行计算求解. 如果采用基于目标函数的评价函数方法，对应每一个 $Y^0 = (y_{it})$ 要解一个线性规划问题. 虽说线性规划问题有多项式时间的最优算法，但对每一个给定的 Y^0 求解一次 $Z(Y^0)$ 还是很费时的. 一个简单的替代方法是对给定的 Y^0，用启发式的算法求解 PP. 基本思想是根据 Y^0、d_{it} 和 c_t 安排各时段的生产(可参见文献[4])，最后得到一个启发式

算法的目标值 $Z^H(\boldsymbol{Y}^0)$，以此为评价函数值.

求解的基本思想是：当没有资源约束(2.3)时，PP模型存在一个最优解满足

$$I_{it-1}x_{it} = 0,$$

即上一个时段有库存，则本时段不生产，反之本时段要生产，则上一个时段必然库存为零；由此对应唯一一个解 $\boldsymbol{Y} \in \{0,1\}^{n\times T}$ 满足

$$x_{i\tau_1} = \begin{cases} \sum_{t=\tau_1}^{\tau_2} d_{it}, y_{i\tau_1} = y_{i\tau_2+1} = 1,\ y_{it} = 0,\ \tau_1 < t \leqslant \tau_2, \\ 0, \qquad 其它. \end{cases} \tag{2.5}$$

当增加(2.3)资源约束后，从最后一个时段 T 开始按资源约束逐个时段验证解(2.5)是否满足(2.3). 当不满足时，将超出资源约束的部分前移一个时段加工. 如此修正，最后若第一个时段资源不足时，则认为无可行解，此时目标值记为充分大的一个数. 若可行，则按修正后的解 $x_{it}, I_{it}, Y_{it}(i = 1,2,\cdots,n;\ t = 1,2,\cdots,T)$，用(2.1)计算目标函数值. 这个目标值作为评价值. □

例 2.11 用一个启发式算法解的目标值替代最优值. 取代的主要原因是认为线性规划算法的计算时间还是太长. 如果计算目标值比较复杂或所花费的时间较长，可以用替代的方法. 应该注意的是替代函数应尽可能地反映原目标函数的特性.

2.3.3 特赦规则

在禁忌搜索算法的迭代过程中，会出现候选集中的全部对象都被禁忌，或有一对象被禁，但若解禁则其目标值将有非常大的下降情况. 在这样的情况下，为了达到全局的最优，我们会让一些禁忌对象重新可选. 这种方法称为特赦，相应的规则称为特赦规则(aspiration criteria). 先用下面的例子理解特赦规则的应用.

例 2.12　五城市的 TSP，其城市间的距离为：

$$D = (d_{ij}) = \begin{bmatrix} 0 & 2 & 10 & 10 & 1 \\ 1 & 0 & 2 & 10 & 10 \\ 10 & 1 & 0 & 2 & 10 \\ 10 & 10 & 1 & 0 & 2 \\ 2 & 10 & 10 & 1 & 0 \end{bmatrix},$$

假设初始解为:$(ACBDE)$. 类似例 2.3,经过一步运算,得到一个解是$(ABCDE)$,目标值为 10,假设在计算的某一步,得到目前的一个解为$(AEDBC)$,$f(AEDBC) = 24$,此时被禁的对换还包括 BC. 若交换 BC,得目标值 5!这个目标值好于前面的任何一个最佳候选解,有理由解禁 BC 对换. 这是一种特赦规则. □

以下罗列三种常用的特赦规则,在下面的讨论中,认为评价值越小越好.

(1) 基于评价值的规则. 例 2.12 就是这样的规则. 在整个计算过程中,记忆已出现的最好解 x^{best}. 当候选集中出现一个解 x^{now},其评价值(可能是目标值)满足 $c(x^{best}) > c(x^{now})$ 时,虽说从 x^{best} 达到 x^{now} 的变化是被禁忌的,此时,解禁 x^{now} 使其自由. 直观理解,我们得到一个更好的解.

(2) 基于最小错误的规则. 当候选集中所有的对象都被禁忌时,而(1)的规则又无法使程序继续下去. 为了得到更好的解,从候选集的所有元素中选一个评价值最小的状态解禁.

(3) 基于影响力的规则. 有些对象的变化对目标值的影响很大,而有的变化对目标值的变化较小. 我们应该关注影响大的变化. 从这个角度理解,如果一个影响大的变化成为被禁对象,我们应该使其自由,这样才能得到问题的一个更好的解. 需要注意的是,我们不能理解为:对象的变化对目标影响大就一定使得目标(或是评价值)变小,它只是一个影响力指标. 这一规则应结合禁忌长度和评价函数值使用. 如在候选集中目标值都不及当前的最好解,而一个禁忌对象的影响指标很高且很快将被解禁时,我们可

以通过解禁这个状态以期望得到更好的解. 下面用 0-1 背包问题的一禁忌搜索算法理解影响力指标.

例 2.13 0-1 背包问题的禁忌搜索算法

解的形式为: $x \in \{0,1\}^n$, 邻域定义为:

$$N: x \to N(x) = \{y \mid \sum_{i=1}^{n} |y_i - x_i| \leqslant 1\}.$$

这样的邻域定义要求每次最多只能有一个变量变化, 从 0 变为 1 或是从 1 变为 0. 直观理解是装一个物品进去或是取一个物品出来. 评价值选择目标值, 此时目标值越大越好. 在计算的每一个迭代过程中, 需要验证能力的可行性, 当所装物品体积大于包的容积时为不可行解, 其目标值定义为 $-K$(K 是一个充分大的整数, 表示得到的解为不可行解). 除了目标值以外, 一个有影响力的指标是物品的大小. 取出一个体积大的物品可以装进多个小的物品, 装进一个大的物品很快使包的容积变小. 假设刚装进一个大的物品, 此时包正好装满, 马上取出是应该禁忌的. 如五个物品的 0-1 背包问题, 它们的体积和价值 $\{a_i, c_i\}$ $(i=1,2,3,4,5)$ 分别为 $\{4,4\}$, $\{2,4\}$, $\{1,1.5\}$, $\{3,4\}$, $\{1.5,4\}$, 包的容积为 6. 假设计算过程中得到一个解为 $x_1 = x_5 = 1$ 和 $x_2 = x_3 = x_4 = 0$, $x_1: 1 \to 0$ 和 $x_5: 1 \to 0$ 受禁. 由于包中不可能再装入任何物品, 为了不使计算停止, 只能采取特赦的方法. 因为体积是能力约束的最重要的指标, 同时对目标函数最具有影响力, 因此选体积大的特赦即 $x_1: 1 \to 0$ 解禁. 这样其他物品就有装包的可能性. □

2.3.4 记忆频率信息

在计算的过程中, 记忆一些信息对解决问题是有利的. 如一个最好的目标值出现的频率很高, 这使我们有理由推测: 现有参数的算法可能无法再得到更好的解, 因为重复的次数过高, 使我们认为可能出现了多次循环. 根据解决问题的需要, 我们可以记忆解集

合、有序被禁对象组、目标值集合等的出现频率. 一般可以根据状态的变化将频率信息分为两类:静态和动态.

静态的频率信息主要是某些变化,诸如解、对换或目标值在计算中出现的频率. 求解它们的频率相对比较简单. 如可以记录它们在计算中出现的次数,出现的次数与总的迭代数的比率,从一个状态出发再回到该状态的迭代次数等. 这些信息有助于我们了解一些解、对换或目标值的重要性,是否出现循环和循环的次数. 在禁忌搜索中,为了更充分的利用信息,一定要记忆目前最优解.

动态的频率信息主要是从一个解、对换或目标值到另一个解、对换或目标值的变化趋势,如记忆一个解序列的变化,或记一个解序列变化的若干个点等. 由于记录比较复杂,因此,它提供的信息量也较大. 在计算动态频率时,通常采用的方法为:

(1) 一个序列的长度,即序列中元素个数. 在记录若干个关键点的序列中,按这些关键点的序列长度的变化进行计算;

(2) 从序列中的一个元素出发,再回到该序列该元素的迭代次数;

(3) 一个序列的平均目标(评价)值,从序列中一个元素到另一个元素目标(评价)值的变化情况;

(4) 该序列出现的频率.

频率信息有助于进一步加强禁忌搜索的效率. 我们可以根据频率信息动态控制禁忌的长度. 当一个元素或一个序列重复出现,我们可以增加禁忌长度以避开循环. 当一个序列的目标(评价)值变化较小时,有必要增加该序列每一个对象的禁忌长度,反之,减少每一个对象的禁忌长度. 一个最佳的目标值出现的频率很高,有理由终止计算而将这一值认为是最优值.

2.3.5　终止规则

无论如何,禁忌搜索算法是一个启发式算法. 我们不可能让禁

忌长度充分大，只希望在可接受的时间里给出一个满意的解. 于是很多直观、易于操作的原则包含在终止规则中. 下面给出常用的终止规则：

(1) 确定步数终止. 给定一个充分大的数 N，总的迭代次数不超过 N 步. 即使算法中包含其他的终止原则，算法的总迭代次数有保证. 这种原则的优点是易于操作和可控计算时间，但无法保证解的效果. 在采用这个规则时，应记录当前最优解.

(2) 频率控制原则. 当某一个解、目标值或元素序列的频率超过一个给定的标准时，如果算法不做改进，只会造成频率的增加，此时的循环对解的改进已无作用，因此，终止计算. 这一规则认为：如果不改进算法，解不会再改进.

(3) 目标值变化控制原则. 在禁忌搜索算法中，提倡记忆当前最优解. 如果在一个给定的步数内，目标值没有改变，同(2)相同的观点，如果算法没有其他改进，解不会改进. 此时，停止运算.

(4) 目标值偏离程度原则. 对一些问题可以简单地计算出它们的下界(目标为极小)，我们在第 6 章将介绍一种求解下界的拉格朗日(Lagrange)松弛算法. 记一个问题的下界为 Z_{LB}，目标值为 $f(x)$. 对给定的充分小的正数 ε，当 $f(x) - Z_{LB} \leqslant \varepsilon$ 时，终止计算. 这表示目前计算得到的解与最优值很接近.

2.4 应用实例

本节介绍两个应用实例，一个是比较简单的图节点着色(node coloring)问题，另一个是车间作业调度，也称为车间作业排序问题. 从两个实例中可以了解禁忌搜索算法应用的一些理论和技术问题.

2.4.1 图节点着色问题

给定一个无向图$G=(V,E)$,其中V是节点集$V=\{1,2,\cdots,n\}$,E是边集,$E=\{(i,j)|i,j\in V\}$,(i,j)表示有连接i,j的一个边. 若$V_i\subset V$、$V=\bigcup_{i=1}^{k}V_i$且V_i内部的任何两个节点没有E中的边直接相连,则称$(V_1,V_2,\cdots,V_k)$为V的一个划分. 图的节点着色问题可以描述为:求一个最小的k,使得$(V_1,V_2,\cdots,V_k)$为V的一个划分.

例 2.14 如图 2.3 所示的五节点无向图$G=(V,E)$. 它的一个划分是:$V_1=\{A,E\}$,$V_2=\{B,D\}$,$V_3=\{C\}$. 当然$V_1=\{A\}$,$V_2=\{B\}$,$V_3=\{C\}$,$V_4=\{D,E\}$也是一个划分. □

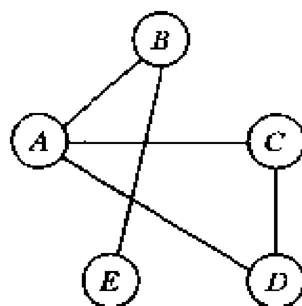


图 2.3 五节点图

对给定n个节点的无向图,禁忌搜索算法求解图节点覆盖问题可以分为两步. 第一步是给定一个常数k,考虑目标函数

$$f(V_1,V_2,\cdots,V_k)=|E(V_1)|+|E(V_2)|+\cdots+|E(V_k)|,$$

其中,$|E(V_i)|$为V_i中直接连接两个节点的边数. $(V_1,V_2,\cdots,V_k)$为V的一个划分的充分必要条件是$f(V_1,V_2,\cdots,V_k)=0$.

第二步的主要工作是:对不满足$f(V_1,V_2,\cdots,V_k)=0$的$(V_1,V_2,\cdots,V_k)$进行优化计算,从而决定是否增加子集的个数k;从满足$f(V_1,V_2,\cdots,V_k)=0$的划分中选择最小的划分数k.

将解的形式用下面形式表示:

$$S=\begin{pmatrix}1 & 2 & \cdots & n\\ i_1 & i_2 & \cdots & i_n\end{pmatrix},\ 1\leqslant i_j\leqslant k,\ 1\leqslant j\leqslant n,$$

其中i_j表示第j个节点在第i_j集合中.

解的第x个分量变化为:

$$\begin{pmatrix} x \\ i_x \end{pmatrix} \rightarrow \begin{pmatrix} x \\ i'_x \end{pmatrix},$$

从原在第 i_x 集合中改变到第 i'_x 集合中.

每一个解 S 的邻域由那些满足上面的变化且只有一个分量变化的解组成，共有 nk 个邻居(包含自身)，每一个节点(分量)可以选择 k 个划分集之一.

禁忌：

$$\begin{pmatrix} x \\ i_x \end{pmatrix} \leftarrow \begin{pmatrix} x \\ i'_x \end{pmatrix},$$

即还原到原有状态. 这种禁忌考虑了变化的方向.

特赦规则：若 x^* 是当前最优解，当一个受禁的邻居 x 满足 $f(x) \leqslant f(x^*) - 1$ 时，则受禁的变化特赦.

因为目标值为非负整数，条件 $f(x) \leqslant f(x^*) - 1$ 表示 x 的目标值小于当前最优解 x^* 的目标值. 本例采用基于评价值的特赦规则(1).

Hertz 和 Werra[5] 在集合的划分数给定的情况下，给出的程序框架如下

已知

划分集合数 k，解的形式 S，目标函数 f，解的变化，邻域映射 $N(S)$，禁忌长度，目标值下界 f^*，两个目标值没有改进的最大允许迭代次数 $nbmax$. 开始

—— 任选一个初始解；

—— $nbiter := 0$;(* 当前的迭代步 *)

—— $bestiter := 0$;(* 当前最优解所在的迭代步 *)

—— $bestsol := S$;(* 当前的最优解 *)

—— 令 $z = f(S)$，$A(z) := z - 1$;(特赦值)

— 初始化禁忌表 H;

当($f(S) > f^*$) 且($nbiter\text{-}bestiter < nbmax$) 时,计算

—$nbiter := nbiter + 1$;

—— 产生 $N(S)$ 的一个候选集 V^*,要求候选元素 x 是非禁忌的或是特赦的 $f(x) \leqslant A(f(S))$;

— 在 V^* 中选目标值最优的解 S^*;

—— 若 $f(S^*) \leqslant A(f(S))$,则 $A(f(S^*)) := f(S^*) - 1$;否则若 $f(S) \leqslant A(f(S^*))$,则 $A(f(S^*)) := f(S) - 1$;

—— 更新禁忌表;

—— 若 $f(S^*) < f(bestsol)$,则 $bestsol := S^*$; $bestiter := nbiter$;

— $S := S^*$

继续;

结束.

输出:计算中的最优解.

在文献[5]中,随机产生实例的参数为:无向图共有1000个节点,边集的密度是50%(大约是 $C_{1000}^2/2$ 个边),禁忌表长度 $|H| = 7$,候选解个数 $|V^*| = |V|/2$. 最终对上面规模实例的计算结果是平均可用87种颜色划分. 概率分析的理论结果是85种颜色.

2.4.2 车间作业调度问题

1. 问题的描述

车间作业调度(job shop scheduling) 问题可以简单地描述为 n 个工件(job)(这是一个统称),$J_1, J_2, \cdots, J_n$ 在 m 台机器(machine)(也是统称)$M_1, M_2, \cdots, M_m$ 上加工. 每一个工件 J_i 有 n_i

个工序(operation),O_{i1},O_{i2},…,O_{in_i},第 O_{ij} 工序的加工时间为 p_{ij}. 必须按工序进行加工且每一工序必须一次加工完成(无抢占,no preemption). 一台机器在任何时刻最多只能加工一个产品,一个工件不能同时在两台机器上加工. 如何在上面的条件下使最后一个完工的工件完工时间最短?

车间作业问题可以用析取图(disjunctive graph)表示:

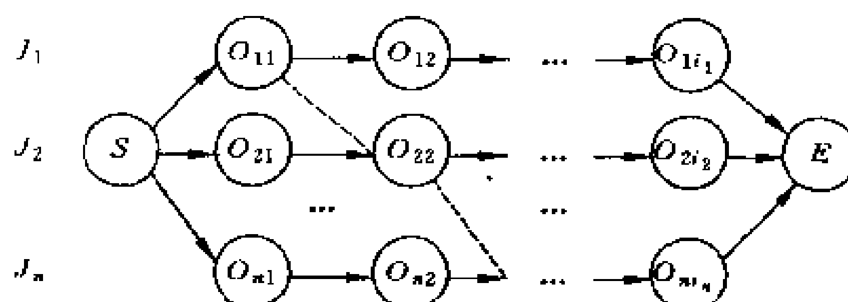


图 2.4　车间作业排序的析取图

在图 2.4 中,S 和 E 是虚拟的起终点,实线弧表示两个工序的前后关系,这是不允许改变的. 每一行表示一个工件的所有工序. 虚线边暂时是没有方向的,一个虚线边表示连接的节点(工序)在同一台机器上加工,如 O_{11},O_{22} 的连接线表示它俩在同一台机器上加工. 车间作业的一个调度是给予所有虚线边一个方向,成为一个虚线弧. 一个虚线弧所指方向表示加工遵照的顺序. 寻找车间作业调度问题的一个可行解就是在图 2.4 中给虚线边赋方向,使其成为一个有向且无圈的图.

车间作业调度问题在两个工件时是存在多项式时间的最优算法,参考文献[6],但当工件数超过 2 时,问题的复杂性是 NP-hard,证明见文献[7]. 非常直观,问题的一个解对应 m 台机器上的一个加工排序.

在应用禁忌搜索之前,由于车间作业的邻域构造有较强的代

表性，我们首先考虑问题的邻域构造. 第一种方法是选一台机器上的两个工序交换位置加工. 如一台机器上原有的加工序为(a,b,c,d,e)，交换ac序后，新的加工序为(c,b,a,d,e). 这样的邻域映射使得一个解有$\sum_{i=1}^{n} C_{n_i}^2$个邻居. 进一步推广上面的方法，可以使得若干个位置交换. 一种特殊的情况是将一个工序移到另一个位置加工. 如在一台机器上原有的加工序是$(abcdef)$，现将f移到第一个位置加工，则加工序为$(fabcde)$. 这些是我们常见的方法.

第二种方法是关键路(critical path)法. 这种方法所依赖的一个基本思想是：在第一种方法中，一些交换对目标值没有影响，这些交换浪费计算时间，应抓住最长的、加工中没有时间空闲的一条路(即关键路)，交换同在这条路上且同在一台机器上加工的两个加工工件的位置.

2. 关键路的理论

例 2.15 三个工件在两台机器上加工的车间作业问题，其加工如图 2.5.

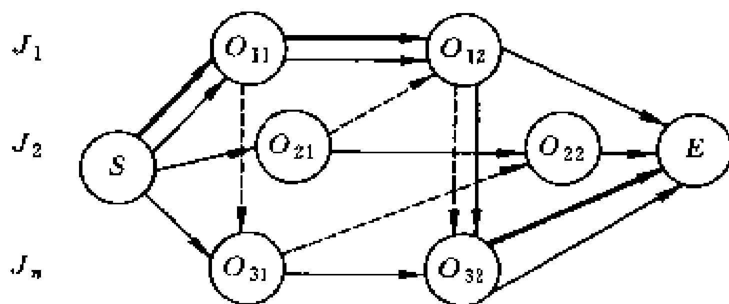


图 2.5 三工件车间作业图

从图中可以看到，O_{11}，O_{22}，O_{31}在一台机器上加工，O_{12}，O_{21}，O_{32}在另一台机器上加工. 假设各工序的加工时间为：$p_{11}=5$，$p_{22}=2$，$p_{31}=4$，$p_{12}=7$，$p_{21}=1$，$p_{32}=2$. 两台机器的加工序分别为：

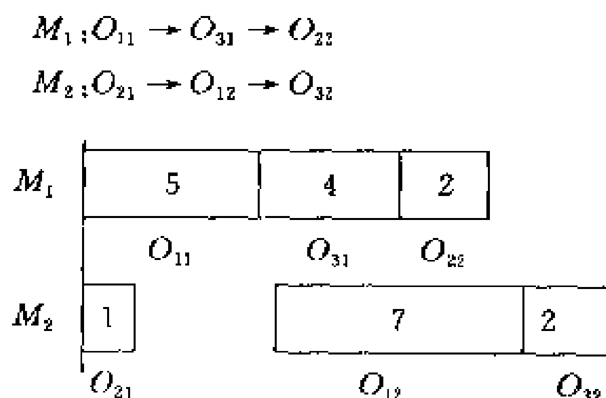


图 2.6　关键路：$O_{11} \to O_{13} \to O_{32}$

关键路是：$O_{11} \to O_{12} \to O_{32}$，长度为 14．以较粗的线标记在图 2.5 上．从图 2.6 可以看出，交换 $O_{31}O_{22}$ 的加工位置对最长完工时间没有影响，但交换关键路上的一个工件，如 $O_{11}O_{31}$ 交换，则如图 2.7．

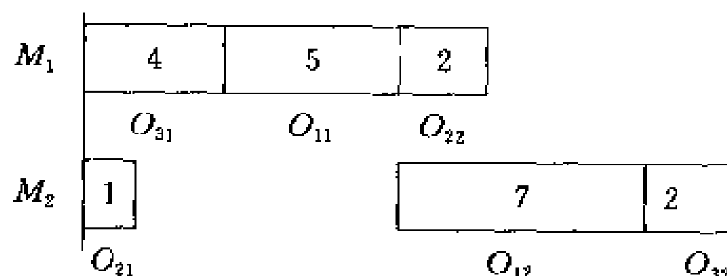


图 2.7　关键路：$O_{31} \to O_{11} \to O_{12} \to O_{32}$

关键路是：$O_{31} \to O_{11} \to O_{12} \to O_{32}$，长度为 18．由此可以看到，关键路上的加工工件的位置改变对目标值有直接影响．　□

定义 2.2　在关键路上，满足下列条件的相邻节点集称为块(block)：

(1) 由关键路上的相邻节点组成，至少包含两个工序；

(2) 集合中的所有工序在一台机器上加工；

(3) 增加一个工序后，不满足(1)或(2)．

定理 2.2　若一解的关键路不包含块，则一定是最优解．

证明 由定理的条件，关键路上不包含块. 由图 2.4，同一台机器的相邻工序由虚线相连，由此可知，任意相邻的两个工序由一条实线连接. 这条关键路长度是某一个工件的所有工序加工时间和，是车间作业完工时间的一个下界. 因此定理结论成立. □

若 y 是车间作业的一个可行解，则对应 y，图 2.4 为一个有向无圈图. 若 y 和 y' 是作业调度的两个可行解，且 y 和 y' 对应的有向图 S 和 S'. 当图 S' 的关键路 P' 的长度小于图 S 的关键路 P 的长度时，称可行解 y' 改进了 y.

定理 2.3[8] 若 y 和 y' 是车间作业调度的两个可行解，且 y 和 y' 对应的有向图为 S 和 S'. 若 y' 改进 y，则一定满足下面的两条件之一：

(1) 至少有一个工序，它在 y 的一个块 B 中且不是 B 块中的第一个工序，但在 y' 中它在 B 的其他工序之前加工；

(2) 至少有一个工序，它在 y 的一个块 B 中且不是 B 块中的最后一个工序，但在 y' 中它在 B 的其他工序之后加工.

证明 y 对应的一条关键路为 P：

$$P: S, u_1^1, \cdots, u_{b_1}^1, u_1^2, \cdots u_{b_2}^2, u_1^k \cdots, u_{b_k}^k, E,$$

其中，$u_1^j, \cdots, u_{b_j}^j$ 表示一个块，即在同一台机器上加工. 现用反证法证明结论. 假设 y' 和 y 的块且所在的机器相同，且 y' 属于 P 的块中的任何一个工序不在其块的其他工序之前或之后加工，则有：

- 在 y' 中，$u_{b_j}^j$ 在 u_1^{j+1} 之前加工，且它们是同一个工件的不同工序；
- 若 $b_j \geq 2$，无论是在 y 还是在 y' 中，$u_1^j, \cdots, u_{b_j}^j$ 在同一台机器上加工；
- 若 $b_j \geq 2$，无论是在 y 还是在 y' 中，u_1^j 总是在 $u_2^j, \cdots, u_{b_j}^j$ 之前加工，$u_{b_j}^j$ 总是在 $u_1^j, \cdots, u_{b_j-1}^j$ 之后加工；

在 y' 中，$u_1^j, \cdots, u_{b_j}^j$ 的加工序为 $v_1^j, \cdots, v_{b_j}^j$，但由上面的讨论，$u_1^j = v_1^j$ 和 $u_{b_j}^j = v_{b_j}^j$. 于是 y' 对应的一条路（不一定为关键路）为 P'：

$$P':S,v_1^1,\cdots,v_{b_1}^1,v_1^2,\cdots,v_{b_2}^2,v_1^k,\cdots,v_{b_k}^k,E,$$

y' 的最大完工时间 $L(y')$ 为：

$$L(y')\geqslant\sum_{j=1}^{k}\sum_{l=1}^{b_j}p_{v_l^j}=\sum_{j=1}^{k}\sum_{l=1}^{b_j}p_{u_l^j}=L(y),$$

y' 不是 y 的一个改进，则与定理的已知条件相矛盾，故定理结论成立. □

3. 邻域结构

通过定理 2.3，可以给出基于关键路的邻域构造方法.

邻域 $N1$：设 y' 为一个可行解，若 y' 将 y 的关键路中一个块中的一个作业前移到最前或后移到最后位置加工，则称 y' 为 y 的一个邻居. 所有这样的移动组成的集合为 y 的邻域.

邻域 $N2$：设 y' 为一个可行解，若 y' 是 y 的关键路中一个块中的作业前移或后移的所有可能位置加工，则称 y' 为 y 的一个邻居. 所有这样的移动组成的集合为 y 的邻域.

注：在上面第一个邻域定义中，我们强调 y' 为一个可行解，这是因为在车间作业调度中，交换两个工序的加工位置可能出现死锁(deadlock) 现象，如图 2.8，原机器的加工序为：

$$M_1:O_{11}\rightarrow O_{22},$$
$$M_2:O_{12}\rightarrow O_{21},$$

现改变第一台机器的加工序为 $M_1:O_{22}\rightarrow O_{11}$，如图 2.8，此时任何一个工件无法开工，因此称为死锁现象.

定理 2.4 车间作业调度的可行解集合相对 $N2$ 是连通的，即以任何一个可行解为起点，可以通过 $N2$ 达到一个全局最优解.

证明参见文献[9].

4. 计算结果

文献[8] 中研究了车间作业的一种更广泛的情形，多功能机器车间作业排序问题. 在他们的模型中，工序 O_{ij} 可由一个机器集

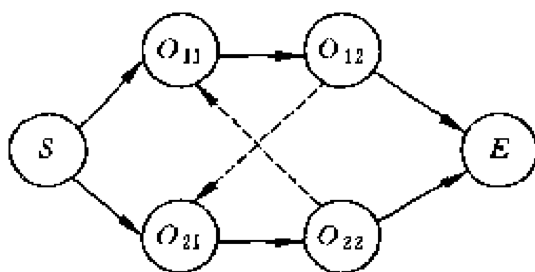


图 2.8 死锁现象

合 $G_{ij} \subset \{M_1, M_2, \cdots, M_m\}$ 中的任何一个机器加工．极端的情况 $|G_{ij}| = 1$ 为常规的车间作业排序问题．他们产生53组随机数据进行数值计算．数据生成的原则是：以文献[7]和文献[10]的典型数据为基础，以表 2.1 的多功能系数分组随机产生数据文件，所产生的数据分别用 edata、rdata 和 vdata 标记．文献[7]和[10]中典型数据的主要参数是：机器数分别为 5、6、10 和 15，工件数分别为 6、10、15、20 和 30，且每一个工件的工序数都相同，满足 $n_i = m, i = 1, 2, \cdots, n$.

表 2.1 多功能参数

	$\|G_{ij}\|$ave	$\|G_{ij}\|$max
edata	1.15	$2(m \leqslant 6)$
		$3(m \geqslant 10)$
rdata	2	3
vdata	$\frac{1}{2}m$	$\frac{4}{3}m$

其中，$|G_{ij}|$ave 表示所有 $|G_{ij}|$ 的平均数，$|G_{ij}|$max 表示所有 $|G_{ij}|$ 的最大数.

用上面的 $N1$ 和 $N2$ 邻域结构，禁忌对象选择为：禁忌上一步位置变化的复原，如一个块为$(abcde)$，c 原在的位置是 3，经过位置移动后，禁忌回到 3. 禁忌表的长度为 30，总的迭代长度分为 1000

和 5000 次两种. 停止规则为:在迭代总长达到之前,若邻域中的所有可行解都是禁忌的或目前解的目标值等于下界或出现循环则停止计算.

功能类型	相对误差	$N1$-1000	$N2$-1000	$N1$-5000	$N2$-5000
edata	平均	5.2	5.3	4.8	4.5
	最大	24.0	22.8	23.4	19.8
rdata	平均	2.8	2.9	2.3	2.3
	最大	13.4	14.5	12.0	10.7
vdata	平均	0.5	0.5	0.4	0.4
	最大	3.2	3.1	1.9	2.1

其中,表中所有数据为禁忌搜索算法所得解的目标值和已知最佳下界的相对误差;$N1$-1000 表示采用 $N1$ 邻域最大迭代次数为 1000;$N2$-5000 表示采用 $N2$ 邻域最大迭代次数为 5000.

计算结果显示,(1) 邻域 $N1$ 和 $N2$ 对计算结果的影响不明显,此处 $N1$ 是否具有定理 2.4 的连通性还没有得到证明;(2) 总体的平均效果,禁忌搜索算法相当好,所得目标值和已知最好下界的相对差的平均效果在 0.4% 到 5.2% 之间. 随着机器功能的增加,相对误差逐渐变小;(3) 迭代次数 1000 和 5000 的差别不是很大,也就是在迭代总次数限制在 1000 次时,计算效果同迭代 5000 次基本相同. 因此可以采用 1000 次的迭代而节省计算时间.

练　习　题

1. 对例 2.13 的五城市 TSP,初始解为($ACBDE$),城市交换对被禁,候选集合采用 2-opt 邻域,禁忌长度为 3,问:算法能否达到最优解($AEDCB$)?在什么样的条件下可以达到最优解?

2. 就禁忌常规状态变化和目标值讨论例 2.13 的收敛情况.

3. 证明:图的节点着色问题是 NP-hard.

4. 在 $N1$ 和 $N2$ 邻域中,在什么条件下,一个块的两个工序位置交换保持可行解?

5. 在练习题 4 的程序实现中,如何判别交换后的解是不可行解?

6. 实现禁忌搜索算法求解图的节点着色问题.

7. 实现禁忌搜索算法求解车间作业调度问题.

8. 实现禁忌搜索算法求解 0-1 背包问题.

参 考 文 献

1. Glover F. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research, 1986, 13:533～549
2. Glover F. Tabu search: part I. ORSA Journal on Computing, 1989,1:190～206
3. Glover F. Tabu search: part II. ORSA Journal on Computing,1990, 2:4～32
4. Maes J, McClain J O, Van Wassenhove L N. Multilevel capacitated lot-sizing complexity and LP-based heuristics. Euro J of Opnl Res, 1991, 53:131～148
5. Hertz A, de Werra D. The tabu search metaheuristic: how we use it. Annals of Mathematics and Artificial Intelligence, 1990, 1:111～121
6. Brucker P, Schilie R. Job-shop scheduling with multi-purpose machines. Computing, 1990, 45:369～375
7. Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job-shop scheduling. Management Sciences, 1988, 34:391～401
8. Hurink J, Jurisch B, Thole M. Tabu search for the job-shop scheduling problem with multipurpose machines. OR Spektrum, 1994, 15:205～215

9. 邢文训. Jobshop 排序问题的模拟退火算法. 见:中国运筹学会第二届全国排序学术会议论文集(武汉),1993. 5~9

10. Fisher H, Thompson G L. Probabilistic learning combinations of local job-shop scheduling Rules. In: Muth J F, Thompson G L, ed. Industrial Scheduling, Englewood Cliffs: Prentice Hall,1963. 225~251

第 3 章

模拟退火算法

模拟退火(simulated annealing)算法是局部搜索算法的扩展.它不同于局部搜索之处是以一定的概率选择邻域中费用值大的状态.理论上来说,它是一个全局最优算法.模拟退火算法最早的思想由 Metropolis[1]在 1953 年提出,Kirkpatrick[2]在 1983 年成功地应用在组合最优化问题中.本章先介绍模拟退火算法,证明它的收敛性,然后讨论算法实现的技术问题,最后用示例理解它的应用.

3.1 模拟退火算法及模型

退火是一种物理过程,一种金属物体在加热至一定的温度后,它的所有分子在状态空间 D 中自由运动.随着温度的下降,这些分子逐渐停留在不同的状态.在温度最低时,分子重新以一定的结构排列.由统计力学的研究表明,在温度 T,分子停留在状态 r 满足波兹曼(Boltzmann)概率分布

$$\Pr\{\overline{E} = E(r)\} = \frac{1}{Z(T)}\exp\left(-\frac{E(r)}{k_B T}\right), \tag{3.1}$$

其中,$E(r)$ 为状态 r 的能量,$k_B > 0$ 为波兹曼常数,$\overline{E}$ 为分子能量的一个随机变量.$Z(T)$ 为概率分布的标准化因子

$$Z(T) = \sum_{s \in D}\exp\left\{-\frac{E(s)}{k_B T}\right\}.$$

先研究由(3.1)确定的函数随 T 变化的趋势.选定两个能量 $E_1 < E_2$,在同一个温度 T,有

$$\Pr(\overline{E}=E_1)-\Pr(\overline{E}=E_2)$$
$$=\frac{1}{Z(T)}\exp\left(-\frac{E_1}{k_BT}\right)\left[1-\exp\left(-\frac{E_2-E_1}{k_BT}\right)\right].$$

因为

$$\exp\left(-\frac{E_2-E_1}{k_BT}\right)<1,\ \forall\, T>0,$$

所以

$$\Pr(\overline{E}=E_1)-\Pr(\overline{E}=E_2)>0,\ \forall\, T>0. \tag{3.2}$$

在同一个温度,(3.2) 表示分子停留在能量小状态的概率比停留在能量大状态的概率要大. 当温度相当高时,(3.1) 的概率分布使得每个状态的概率基本相同,接近平均值 $1/|D|$,$|D|$ 为状态空间 D 中状态的个数. 结合(3.2),具有最低能量状态的波兹曼概率接近并超出平均值 $1/|D|$. 由

$$\frac{\partial\Pr\{\overline{E}=E(r)\}}{\partial T}$$
$$=\frac{\exp\left\{-\frac{E(r)}{k_BT}\right\}}{Z(T)k_BT^2}\left(E(r)-\frac{\sum\limits_{s\in D}E(s)\exp\left\{-\frac{E(s)}{k_BT}\right\}}{Z(T)}\right), \tag{3.3}$$

当 $r_{\min}$ 是 D 中具有最低能量的状态时,得

$$\frac{\partial\Pr\{\overline{E}=E(r_{\min})\}}{\partial T}<0,$$

所以,$\Pr\{\overline{E}=E(r_{\min})\}$ 关于温度 T 是单调下降的. 又有

$$\Pr\{\overline{E}=E(r_{\min})\}=\frac{1}{Z(T)}\exp\left(-\frac{E(r_{\min})}{k_BT}\right)=\frac{1}{|D_0|+R},$$

其中,D_0 是具有最低能量的状态集合,

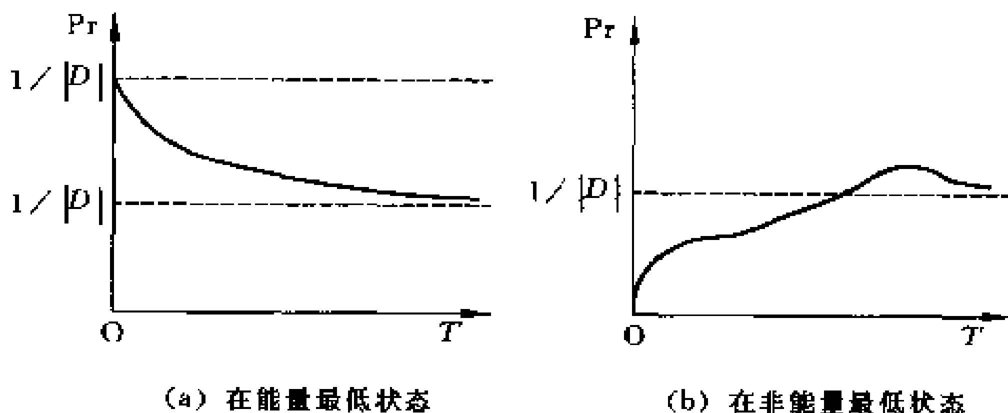
$$R=\sum_{s\in D,E(s)>E(r_{\min})}\exp\left(-\frac{E(s)-E(r_{\min})}{k_BT}\right)\to 0,\ T\to 0. \tag{3.4}$$

因此得到，当 T 趋向于 0 时，

$$\Pr\{\overline{E} = E(r_{\min})\} \to \frac{1}{|D_0|},\ T \to 0.$$

当温度趋向 0 时，(3.1) 决定的概率渐近 $1/|D_0|$. 由此可以得到，在温度趋向 0 时，分子停留在最低能量状态的概率趋向 1. 综合上面的讨论，分子在能量最低状态的概率变化趋势由图 3.1(a) 表示.

对于非能量最小的状态，由 (3.2) 和分子在能量最小状态的概率是单调减小的事实，在温度较高时，分子在这些状态的概率在 $1/|D|$ 附近，依赖于状态的不同，可能超过 $1/|D|$；由 (3.3) 和 (3.4) 可知存在一个温度 t，使 (3.1) 决定的概率在 $(0,t)$ 是单调升的(留作练习)；再由 (3.4) 可知，当温度趋于 0 时，(3.1) 定义的概率趋于 0. 概率变化曲线见图 3.1(b).



(a) 在能量最低状态　　(b) 在非能量最低状态

图 3.1　波兹曼函数曲线

从上面的讨论得到，在温度很低时 $(T \to 0)$，能量越低的状态的概率值越高. 在极限状况，只有能量最低的点概率不为零. 可以从下面的例子了解 (3.1) 的变化规律.

例 3.1　简化概率分布 (3.1) 为

$$p(x) = \frac{1}{q(t)}\exp\left(-\frac{x}{t}\right),$$

其中，$q(t)$ 为标准化因子. 设共有四个能量点 $x = 1,2,3,4$，在此 $q(t) = \sum_{x=1}^{4}\exp\left(-\frac{x}{t}\right)$. 观察 $t = 20,5,0.5$ 三个温度点概率分布变化.

从表 3.1 可以看到，当温度较高时（$t = 20$），四点的概率分布相差比较小，也可以看成概率是均匀分布，但能量最低状态 $x = 1$ 的概率 0.269 超出平均值 0.25. 这相当于分子的随机游动. 当温度下降后（$t = 5$），状态 $x = 4$ 的发生概率变得比较小了，也就是说，它的活跃程度下降. 当 $t = 0.5$ 时，$x = 1$ 的概率达 0.865，而其他三个状态的概率都很小，合起来为 0.135. 在这个表中也可以看出，非能量最低状态 $x = 2$ 的概率在三个温度点（0.5，5，20）有一个上升和下降的过程，在 $t = 20$ 和 5 时的概率都超过平均概率 0.25. □

表 3.1　$t = 20,5,0.5$ 三个温度点的概率分布

	$x = 1$	$x = 2$	$x = 3$	$x = 4$
$t = 20$	0.269	0.256	0.243	0.232
$t = 5$	0.329	0.269	0.221	0.181
$t = 0.5$	0.865	0.117	0.016	0.002

通过这个示例，我们可以将组合优化问题同金属物体退火进行类比：

组合优化问题	金属物体
解	状态
最优解	能量最低的状态
费用函数	能量

由以上的类比及(3.1)，组合优化的最优解可以类比为退火过程中能量的最低状态，也就是温度达到最低点时，(3.1) 概率分布

中具有最大概率的状态.于是组合优化问题 $z = \min\{f(\boldsymbol{x}) \mid g(\boldsymbol{x}) \geqslant 0, \boldsymbol{x} \in D\}$ 的求解过程类比为退火过程.其中 D 是有限离散定义域.

在这一章中,除特别强调外,我们都假设算法用以解决如下组合最优化问题:

$$\begin{aligned} &\min f(\boldsymbol{x}) \\ &\text{s.t. } g(\boldsymbol{x}) \geqslant 0, \\ &\quad \boldsymbol{x} \in D. \end{aligned}$$

其中 $f(\boldsymbol{x})$ 为目标函数,$g(\boldsymbol{x})$ 为约束方程,D 为定义域.

简单的模拟退火算法为:

模拟退火算法

STEP1 任选一个初始解 $\boldsymbol{x}_0$;$\boldsymbol{x}_i := \boldsymbol{x}_0$;$k := 0$;$t_0 := t\max$(初始温度);

STEP2 若在该温度达到内循环停止条件,则到 STEP3;否则,从邻域 $N(\boldsymbol{x}_i)$ 中随机选一 $\boldsymbol{x}_j$,计算 $\triangle f_{ij} = f(\boldsymbol{x}_j) - f(\boldsymbol{x}_i)$;若 $\triangle f_{ij} \leqslant 0$,则 $\boldsymbol{x}_i := \boldsymbol{x}_j$,否则若 $\exp(-\triangle f_{ij}/t_k) >$ random(0,1) 时,则 $\boldsymbol{x}_i := \boldsymbol{x}_j$;重复 STEP2;

STEP3 $t_{k+1} := d(t_k)$;$k := k+1$;若满足停止条件,终止计算;否则,回到 STEP2.

在上述的模拟退火算法中,包含一个内循环和一个外循环.内循环是 STEP2,它表示在同一个温度 t_k 时,在一些状态随机搜索.外循环主要包括 STEP3 的温度下降变化 $t_{k+1} := d(t_k)$,迭代步数的增加 $k := k+1$ 和停止条件.这些内容将在 3.3 节中讨论.模拟退火的直观理解是:在一个给定的温度,搜索从一个状态随机地变化到另一个状态.每一个状态到达的次数服从一个概率分布.当温度很低时,由(3.4)的讨论,以概率 1 停留在最优解.这些理论将在

3.3 节介绍.

模拟退火算法的数学模型可以描述为:在给定邻域结构后,模拟退火过程是从一个状态到另一个状态不断地随机游动.我们可以用马尔可夫(Markov)链描述这一过程.当温度 t 为一确定值时,两个状态的转移概率(transition probability)定义为:

$$p_{ij}(t)=\begin{cases}G_{ij}(t)A_{ij}(t), & \forall j\neq i,\\ 1-\sum\limits_{l=1,l\neq i}^{|D|}G_{il}(t)A_{il}(t), & j=i,\end{cases}\tag{3.5}$$

其中,$|D|$ 表示状态集合(解集合)中状态的个数.$G_{ij}(t)$ 称为从 i 到 j 的产生概率(generation probability).$G_{ij}(t)$ 表示在状态 i 时,j 状态被选取的概率.比较容易理解的是 j 是 i 的邻居,如果在邻域中等概率选取,则 j 被选中的概率为

$$G_{ij}(t)=\begin{cases}1/|N(x_i)|, & j\in N(i),\\ 0, & j\notin N(i).\end{cases}\tag{3.6}$$

$A_{ij}(t)$ 称为接受概率(acceptance probability).$A_{ij}(t)$ 表示在状态 i 产生 j 后,接受 j 的概率,如在模拟退火算法中接受的概率是

$$A_{ij}(t)=\begin{cases}1, & f(i)\geqslant f(j),\\ \exp(-\Delta f_{ij}/t), & f(i)<f(j).\end{cases}\tag{3.7}$$

$G(t)=(G_{ij}(t))_{|D|\times|D|}$,$A(t)=(A_{ij}(t))_{|D|\times|D|}$ 和 $P(t)=(p_{ij}(t))_{|D|\times|D|}$ 分别称为产生矩阵,接受矩阵和一步转移概率矩阵.

(3.5),(3.6),(3.7) 为模拟退火算法的主要模型.模拟退火算法主要可以分为两类.第 1 类为时齐算法,在(3.5) 中对每一个固定的 t,计算对应的马尔可夫链变化,直至达到一个稳定状态,然后再使温度下降.第 2 类是非时齐算法,由一个马尔可夫链组成,要求在两个相邻的转移中,温度 t 是下降的.

无论从实际和直观,描述模拟退火过程的马尔可夫链应满足下列条件:

(1) 可达性. 无论起点如何,任何一个状态都可以到达. 这样使我们有得到最优解的可能,否则,从理论上无法达到最优解的算法是无法采用的.

(2) 渐近不依赖起点. 由于起点的选择有非常大的随机性,我们的目的是达到全局最优,因此,应渐近的不依赖起点.

(3) 分布稳定性. 包含两个内容:一是当温度不变时,其马尔可夫链的极限分布存在;二是当温度渐近 0 时,其马尔可夫链也有极限分布.

(4) 收敛到最优解. 当温度渐近 0 时,最优状态的极限分布和为 1.

下一节将从理论上讨论这些要求能否达到.

3.2 马尔可夫链

设 $\{X(k)\}_{k=1,2,\cdots}$ 为随机变量序列, $X(k)=i$ 称在时刻 k 处于状态 i, $i\in\Omega$. Ω 称为状态空间. 当 Ω 中的状态数有限时,称为有限状态空间. 若对任意的正整数 n, $\{X(k)\}_{k=1,2,\cdots}$ 满足

$$\begin{aligned}\Pr\{X(n)=j\,|\,X(1)=i_1,X(2)=i_2,\cdots,X(n-2)\\=i_{n-2},X(n-1)=i\}\\=\Pr(X(n)=j\,|\,X(n-1)=i),\\i_1,i_2,\cdots,i_{n-2},i,j\in\Omega.\end{aligned}\tag{3.8}$$

则称 $\{X(k)\}_{k=1,2,\cdots}$ 为马尔可夫链,简称马氏链. 马氏链具有一种记忆遗忘功能,它只记忆前一时刻的状态. (3.8) 可以简单地记成

$$p_{ij}(n-1)=\Pr(X(n)=j\,|\,X(n-1)=i),$$

称为一步转移概率. 当状态空间有限时,称为有限马氏链,一步转移概率矩阵记为

$$\boldsymbol{P}(n-1)=(p_{ij}(n-1))_{|\Omega|\times|\Omega|}.$$

当

$$p_{ij}(n-1)=p_{ij}(n),\ \forall\ n, \tag{3.9}$$

成立时，称马氏链为时齐(homogeneous)的. 用

$$p_{ij}^{(n)}=\Pr(X(n)=j|X(0)=i), \tag{3.10}$$

表示 n 步转移概率.

例 3.2 当温度 t 给定时，由(3.5),(3.6),(3.7)确定的马氏链是时齐的.

由(3.5),(3.6),(3.7)可以看出，一步转移概率只和状态 i 转移到 j 有关，同第几次迭代无关. 因此，确定的马氏链是时齐的. 正是这个原因将这一类算法取名为时齐算法. □

在讨论算法的收敛性之前，先介绍概率论中的常用概念.

若存在 n，使得 $p_{ij}^{(n)}>0$，则称状态 i 可达状态 j，记成 $i\rightarrow j$. 若状态 i 和状态 j 满足 $i\rightarrow j$ 且 $j\rightarrow i$，则称状态 i 和状态 j 相通，记成 $i\leftrightarrow j$. 可达关系具有半序性(semi-order).

定理 3.1 若 $i\rightarrow j, j\rightarrow k$，则 $i\rightarrow k$.

证明 由 $i\rightarrow j$，则存在 n，使得 $p_{ij}^{(n)}>0$. 由 $j\rightarrow k$，则存在 m，使得 $p_{ij}^{(m)}>0$. 于是，

$$\begin{aligned}
p_{ik}^{(m+n)} &= \Pr(X(n+m)=k|X(0)=i)\\
&= \sum_{l=0}^{\infty}\Pr(X(m-n)=k, X(n)=l|X(0)=i)\\
&= \sum_{l=0}^{\infty}\Pr(X(m+n)=k|X(n)=l, X(0)=i)\Pr(X(n)=l|X(0)=i)\\
&= \sum_{l=0}^{\infty}\Pr(X(m)=k|X(0)=l)\Pr(X(n)=l|X(0)=i)\\
&\geqslant p_{ij}^{(n)}p_{jk}^{(m)}>0,
\end{aligned}$$

所以，定理成立. □

定义从 i 到达 j 的首达时刻随机变量为

$$T_{ij}=\min\{n|X(0)=i, X(n)=j, n\geqslant 1\}, \tag{3.11}$$

其概率定义为

$$
\begin{aligned}
f_{ij}^{(n)} &= \Pr(T_{ij} = n \mid X(0) = i) \\
&= \Pr(X(n) = j, X(m) \neq j, m = 1,2,\cdots,n-1 \mid X(0) = i).
\end{aligned} \tag{3.12}
$$

迟早到达概率定义为

$$
f_{ij} = \sum_{n=1}^{\infty} f_{ij}^{(n)}. \tag{3.13}
$$

定理 3.2 $f_{ij} > 0$ 的充分必要条件是 $i \to j$.

证明 "充分性":由 $i \to j$,则存在 n,使得 $p_{ij}^{(n)} > 0$. 又

$$
\begin{aligned}
p_{ij}^{(n)} &= \Pr\{X(n) = j \mid X(0) = i\} \\
&= \Pr\{T_{ij} \leqslant n, X(n) = j \mid X(0) = i\} \\
&= \sum_{l=1}^{n} \Pr\{T_{ij} = l, X(n) = j \mid X(0) = i\} \\
&= \sum_{l=1}^{n} \Pr\{T_{ij} = l \mid X(0) = i\} \Pr\{X(n) = j \mid T_{ij} = l, X(0) = i\} \\
&= \sum_{l=1}^{n} \Pr\{T_{ij} = l \mid X(0) = i\} \Pr\{X(n) = j \mid X(l) = j, X(l-1) \\
&\qquad \neq j, \cdots, X(1) \neq j, X(0) = i\} \\
&= \sum_{l=1}^{n} \Pr\{T_{ij} = l \mid X(0) = i\} \Pr\{X(n) = j \mid X(l) = j\} \\
&= \sum_{l=1}^{n} f_{ij}^{(l)} p_{jj}^{(n-l)},
\end{aligned}
$$

当 $p_{ij}^{(n)} > 0$ 时,至少有一个 l 使得 $f_{ij}^{(l)} > 0$,由此,得 $f_{ij} > 0$.

"必要性": 若 $f_{ij} > 0$,则存在 n,使得 $f_{ij}^{(n)} > 0$,得到

$$
p_{ij}^{(n)} \geqslant f_{ij}^{(n)} p_{jj}^{(0)} = f_{ij}^{(n)} > 0。
$$

□

当 $f_{ii} = 1$ 时,状态 i 称为常返的,当 $f_{ii} < 1$ 时称为非常返的.

定理 3.3 状态 j 是常返的,则以概率 1,系统无穷次返回状态 j. 状态 j 是非常返的,则以概率 1,系统只有有限次返回状态 j.

证明　记 $A_{ij}(m)$ 表示自状态 i 出发，系统通过 j 状态至少 m 次的概率. 记 $Y_{ij}(m)$ 为从状态 i 出发通过 j 状态至少 m 次的随机变量.

$$\begin{aligned} A_{ij}(m+1) &= \sum_{l=1}^{\infty}\Pr(T_{ij}=l, Y_{jj}(m) \mid X(0)=i) \\ &= \sum_{l=1}^{\infty}\Pr(Y_{jj}(m) \mid T_{ij}=l)\Pr(T_{ij}=l \mid X(0)=i) \\ &= \sum_{l=1}^{\infty}A_{jj}(m)f_{ij}^{(l)} \\ &= A_{jj}(m)f_{ij}, \end{aligned}$$

于是有，$A_{jj}(m) = A_{jj}(0)(f_{jj})^m = f_{jj}^m$，进而，$A_{ij}(m+1) = f_{ij}(f_{jj})^m$，所以，

$$A_{ij}(m+1) \to \begin{cases} 1, & f_{jj}=1, \\ 0, & f_{jj}<1, \end{cases} (m \to \infty).$$ □

常返中的一种特殊情况为正常返，定义

$$u_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)}, \tag{3.14}$$

当 $u_i < \infty$ 时为正常返，当 $u_i = \infty$ 时为零常返. 定理 3.3 表明常返是以概率 1 无穷次返回同一状态，(3.14) 表明有些常返状态（正常返）的平均返回次数是有限的，而有些是无穷的. 一个状态 i 称为周期(periodic) t (t 正整数)的，如果 $p_{ii}^{(n)}$ 除 $n = t, 2t, 3t, \cdots$ 外均为 0，且 t 是满足这个条件的最大整数. 不存在周期的状态称为非周期状态. 非周期的正常返状态称为遍历状态.

在模拟退火的理论中，经常用到的一个概念是不可约(irreducible). 不可约中用到的一个概念是闭集. 一个集合 C 是闭集的定义为：$\forall\, i \in C, j \notin C$，有 $p_{ij} = 0$. 这表示 C 是一个封闭的集合 $\forall\, i \in C, j \notin C$，$i$ 不可达 j，即 $p_{ij}^{(n)} = 0$，对任意 n 成立. 除整个状态空间外，没有别的闭集的马氏链称为不可约的马氏链. 我们不加

证明地给出下面重要的结论,定理 3.4 和定理 3.5.相应的结论和证明可参见随机过程的教科书.

定理 3.4 不可约的有限状态时齐的马氏链都是正常返的.

模拟退火中用到的一个重要分布是平稳分布(stationary distribution).一个概率分布$\{v_j | j = 1,2,\cdots\}$,即$v_j \geqslant 0, j = 1,2,\cdots$,$\sum_{j=1}^{\infty} v_j = 1$,如果对马氏链的一步转移概率阵$P = (p_{ij})$满足

$$v_j = \sum_{i=1}^{\infty} v_i p_{ij}, \tag{3.15}$$

则称$\{v_j | j = 1,2,\cdots\}$为马氏链的平稳分布.

定理 3.5 非周期不可约时齐马氏链是正常返的充分必要条件是存在唯一平稳分布$\{v_j | j = 1,2,\cdots\}$,满足

$$v_j = \sum_{i=1}^{\infty} v_i p_{ij}^{(n)} = \sum_{i=1}^{\infty} v_i p_{ij},$$

$$v_j = \lim_{n\to\infty} p_{ij}^{(n)} = \frac{1}{u_j} > 0,$$

其中u_j由(3.14)定义给出.

由不可约和闭集的定义,可以得到马氏链是不可约的一个充分条件,用下面的定理叙述:

定理 3.6 有限状态时齐的马氏链是不可约的一个充分条件为:任给两个状态i和j,存在n,使得$p_{ij}^{(n)} > 0$.

证明 由条件,状态空间的任何一个真子集不形成闭集.反证法.若形成闭集,则存在C和$j \notin C$,使得$\forall i \in C, j \notin C$,$i$与$j$是不可达的,即$p_{ij}^{(n)} = 0$,对任意$n$成立.这与条件矛盾.因此,整个状态空间为一个闭集,是不可约的. □

在模拟退火中,要用到的一个重要的结论是定理 3.5.定理 3.6对有限状态的马氏链给出不可约的一个充分条件.定理 3.5 中用到的另一个概念是非周期.由下面的定理,给出判别非周期的一

个充分条件.

定理3.7 若i和j相通,即$i\leftrightarrow j$,则它们或同为非周期的或同为周期的.

证明 由$i\leftrightarrow j$,存在$n\geqslant 1$和$l\geqslant 1$,使得$p_{ij}^{(n)}=a>0$和$p_{ji}^{(l)}=b>0$. 类似定理3.1的证明,可得:

$$p_{ii}^{(l+m+n)}\geqslant p_{ij}^{(n)}p_{jj}^{(m)}p_{ji}^{(l)}=abp_{jj}^{(m)}, \tag{3.16}$$

$$p_{jj}^{(l+m+n)}\geqslant p_{ji}^{(l)}p_{ii}^{(m)}p_{ij}^{(n)}=abp_{ii}^{(m)}. \tag{3.17}$$

再由周期的定义,若i的周期为t,当m是t的倍数时,有$p_{ii}^{(m)}>0$. 再由(3.16),得到

$$p_{ii}^{(l+n)}\geqslant p_{ij}^{(n)}p_{jj}^{(0)}p_{ji}^{(l)}=ab>0,$$

则$l+n$是t的倍数,且由(3.17)得$p_{jj}^{(l+m+n)}>0$. 当m不是t的倍数时,因$l+n$是t的倍数,得$l+m+t$不是t的倍数,由(3.16)得$p_{jj}^{(m)}=0$. 总结上面的讨论,$l+n$是t的倍数,当m不是t的倍数时得$p_{jj}^{(m)}=0$,故状态j是周期的,其周期不小于t. 同法,当j是周期的,可知i是周期的. 于是,它们或同为周期的或同为非周期的. □

模拟退火算法可以分为时齐(homogeneous)的和非时齐(inhomogeneous)的. 由以上马氏链性质的讨论,收敛性证明沿用下面过程.

时齐算法:

(1) 在每一个给定的温度t,给出(3.5)的一步转移概率$p_{ij}(t)$的一些限定条件,如$p_{ij}(t)$满足定理3.6和定理3.7的条件,使得定理3.5成立. 由此得到平稳分布概率

$$v_j(t)=\lim_{n\to\infty}p_{ij}^{(n)}(t)=\lim_{n\to\infty}\Pr(X(n,t)=j\,|\,X(0,t)=i)=\frac{1}{u_j}>0,$$

其中,$X(n,t)$表示在温度t时,马氏链第n步运动的随机变量.

(2) 给出平稳分布应该满足的条件,使得:当温度渐近达到零度时,平稳分布的极限存在,即要求$\pi_j=\lim_{t\to 0}v_j(t)$.

(3) 进一步要求平稳分布的极限具有全局最优性条件

$$\pi_j = \begin{cases} \dfrac{1}{|D_{\mathrm{OPT}}|}, & j \in D_{\mathrm{OPT}}, \\ 0, & j \in D \setminus D_{\mathrm{OPT}}, \end{cases}$$

其中 D_{OPT} 为最优状态集合.

综合上面 3 条,得到全局性收敛的表达式

$$\lim_{t\to 0}\{\lim_{n\to\infty}\Pr(X(n,t) \in D_{\mathrm{OPT}})\} = 1.$$

对非时齐算法,也需要下面的条件保证其全局收敛性.

(1) 因为非时齐算法的每步迭代温度在变化,因此需要给出(3.5)一步转移概率 $p_{ij}(t)$ 的一些限定条件,即需要给出 $A(t_k)$ 和 $G(t_k)$ 的限定条件,使得非时齐马氏链能收敛到一个分布.

(2) 温度是渐近达到零度的,即 $\lim_{k\to\infty} t_k = 0$. 给出保证达到全局最优的温度变化关系.

(3) 最终要求上面(1)和(2)的限定条件使得 $\lim_{k\to\infty}\Pr\{X(k,t_k) \in D_{\mathrm{OPT}}\} = 1$.

3.3 时齐算法的收敛性

由于组合最优化问题的状态空间是有限的,要达到定理 3.5 的条件,由定理 3.4 只需讨论模拟退火对应的马氏链(3.5)的不可约性和非周期性即可. 由定理 3.6,有限状态时齐的马氏链是不可约的一个充分条件为:任给两个状态 i 和 j,存在 n,使得 $p_{ij}^{(n)} > 0$. 于是

定理 3.8 时齐算法的马氏链是不可约的一个充分条件为:

$$\forall\, i,j,t,\text{有 } A_{ij}(t) > 0,$$

且存在 $q \geq 1$, $l_0, l_1, \cdots, l_q \in D$, $l_0 = i$, $l_q = j$,使得

$$G_{l_k l_{k+1}}(t) > 0,\ k = 0,1,\cdots,q-1.$$

证明 当 t 是给定时,由于(3.5)是有限的时齐马氏链,又由

$$\begin{aligned}p_{ij}^{(q)} &\geqslant p_{l_0l_1}p_{l_1l_2}\cdots p_{l_{q-1}l_q} \\ &= A_{l_0l_1}(t)G_{l_0l_1}(t)A_{l_1l_2}(t)G_{l_1l_2}(t)\cdots A_{l_{q-1}l_q}(t)G_{l_{q-1}l_q}(t) \\ &> 0,\end{aligned}$$

所以，i 可达 j. 同理可证 j 可达 i. 于是再由定理 3.6，证明不可约结论成立. □

定理 3.9　对给定的 $t>0$，若存在 $i\neq j\in D$，使得 $A_{ij}(t)<1, G_{ij}(t)>0$，则不可约的马氏链是非周期的.

证明　由

$$\begin{aligned}p_{ii}(t) &= 1-\sum_{l=1,l\neq i}^{|D|}A_{il}(t)G_{il}(t) \\ &= 1-\sum_{l=1,l\neq i,j}^{|D|}A_{il}(t)G_{il}(t)-A_{ij}(t)G_{ij}(t) \\ &> 1-\sum_{l=1,l\neq i,j}^{|D|}G_{il}(t)-G_{ij}(t) \\ &= 1-\sum_{l=1,l\neq i}^{|D|}G_{il}(t) \\ &= G_{ii}(t)\geqslant 0,\end{aligned}$$

此时，$p_{ii}(t)>0$，于是，对任意状态 i 是非周期的. 再由定理 3.7，知结论成立. □

定理 3.8 和定理 3.9 给出存在平稳分布的一个充分条件. 进一步，我们将讨论(3.5) 中 $A(t)$ 和 $G(t)$ 所满足的条件，使得定理 3.5 的平稳分布可以解析地表达.

定理 3.10　若(3.5) 中 $A(t)$ 和 $G(t)$ 满足：

(1) $G(t)$ 与 t 无关；

(2) $\forall\, i,j\in D$，都有 $G_{ij}=G_{ji}$ 且存在 $q\geqslant 1$，$l_0,l_1,\cdots,l_q\in D$，$l_0=i,l_q=j$，使得

$$G_{l_kl_{k+1}}(t)>0,\ k=0,1,\cdots,q-1;$$

(3) $\forall\, i,j,k \in D,\ f(i) \leqslant f(j) \leqslant f(k)$，都有 $A_{ik}(t) = A_{ij}(t)A_{jk}(t)$；

(4) $\forall\, i,j \in D,\ f(i) \geqslant f(j)$，都有 $A_{ij}(t) = 1$；

(5) $\forall\, i,j \in D,\ t > 0,\ f(i) < f(j)$，都有 $0 < A_{ij}(t) < 1$；

则模拟退火时齐算法的马氏链有平稳分布 $v = (v_1, v_2, \cdots, v_{|D|})$，满足

$$v_i(t) = \frac{A_{i_0 i}(t)}{\sum_{j \in D} A_{i_0 j}(t)},\ \forall\, i \in D, \tag{3.18}$$

其中，$i_0 \in D_{\mathrm{OPT}}$.

证明　由(4)和(5)得到 $\forall\, i,j,t$ 有 $A_{ij}(t) > 0$. 再由(1)、(2)和定理 3.8 知时齐算法的马氏链是不可约的. 当所有函数值相同时，每一个解为最优解. 若目标值不是同一个值，选一个最优解和一个非优解，由(2)和(5)知定理 3.9 的条件成立，推出有限时齐的马氏链是非周期的. 因此，由定理 3.5 知平稳分布存在且唯一. 现只需验证(3.18)的概率分布满足平稳分布的定义. 由

$$\begin{aligned}
&\sum_j v_j(t) p_{ji}(t) \\
&= \sum_{j \neq i, f(j) \leqslant f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t) + \sum_{j \neq i, f(j) > f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ji} A_{ji}(t) \\
&\quad + v_i(t) p_{ii}(t) \\
&= \sum_{j \neq i, f(j) \leqslant f(i)} \frac{1}{K} A_{i_0 i}(t) G_{ij} + \sum_{j \neq i, f(j) > f(i)} \frac{1}{K} A_{i_0 j}(t) G_{ij} + v_i(t) p_{ii}(t) \\
&= v_i(t) \sum_{j \neq i, f(j) \leqslant f(i)} G_{ij} + \sum_{j \neq i, f(j) > f(i)} v_j(t) G_{ij} + v_i(t) p_{ii}(t),
\end{aligned}$$

其中，$K = \sum_{j \in D} A_{i_0 j}(t)$，和

$$\begin{aligned}
&v_i(t) p_{ii}(t) \\
&= v_i(t)\Big(1 - \sum_{j \neq i, f(j) \leqslant f(i)} A_{ij}(t) G_{ij} - \sum_{j \neq i, f(j) > f(i)} A_{ij}(t) G_{ij}\Big)
\end{aligned}$$

$$= v_i(t) - v_i(t) \sum_{j \neq i, f(j) \leqslant f(i)} G_{ij} - \sum_{j \neq i, f(j) > f(i)} \frac{1}{K} A_{i_0 i}(t) A_{ij}(t) G_{ij}$$

$$= v_i(t) - v_i(t) \sum_{j \neq i, f(j) \leqslant f(i)} G_{ij} - \sum_{j \neq i, f(j) > f(i)} v_j(t) G_{ij},$$

推出

$$v_i(t) = \sum_j v_j(t) p_{ji}(t),$$

即满足定理 3.5 的平稳分布方程. 由定理 3.5 的唯一性结论,知定理结论成立. □

很容易验证,(3.7) 满足定理 3.10 的(3)(4)(5). (2) 要求任何两个状态或互为邻居或互不为邻居,当互为邻居时,它们的产生概率相同. 当任何两个状态或互为邻居或互不为邻居时,如下面的产生概率:

$$G_{ij}(t) = \begin{cases} \frac{1}{L}, & \forall j \in N(i), \\ 0, & j \notin N(i), \end{cases}$$

满足(1) 和(2) 的 $G_{ij} = G_{ji}$,其中 L 为一正数. (2) 的其他条件需要邻域是连通的. (3.6) 形式对应的结论将在以后的定理中叙述.

定理 3.11 在满足定理 3.10 的条件下,当满足

$$\forall i, j \in D,\ f(i) < f(j) \Rightarrow \lim_{t \to 0} A_{ij}(t) = 0$$

时,则有

$$\lim_{t \to 0} v_i(t) = \begin{cases} \frac{1}{|D_{OPT}|}, & i \in D_{OPT}, \\ 0, & i \in D \setminus D_{OPT}. \end{cases}$$

证明 由(3.18)

$$v_i(t) = \frac{A_{i_0 i}(t)}{\sum_{j \in D} A_{i_0 j}(t)}, \quad \forall i \in D,$$

因为 D 为有限集,故当 $i \notin D_{OPT}$ 时,有

$$f(i) < f(i_0), \quad i_0 \in D_{OPT},$$

于是，

$$A_{i_0 i}(t) \to 0, \quad t \to 0.$$

所以结论成立. □

定理 3.10 和定理 3.11 给出了时齐算法的全局收敛应满足的充分条件，但这些条件的限制是比较强的，如(3.6)的发生概率就不一定满足定理 3.10 的(2)，研究模拟退火算法全局收敛所满足的充分条件是理论研究的一个方向.

定理 3.12 若模拟退火时齐算法中的 $A_{ij}(t)$ 和 $G_{ij}(t)$ 满足定理 3.8 和定理 3.9 的条件，且存在 $\Phi(x,t)$ 满足以下条件：

(1) $\forall i \in D,\ t > 0$，都有 $\Phi(f(i),t) > 0$；

(2) $\forall i \in D$，都有 $\sum_{i=1,i\neq j}^{|D|} \Phi(f(i),t) G_{ij}(t) A_{ij}(t) = \Phi(f(j),t) \sum_{i=1,i\neq j}^{|D|} G_{ji}(t) A_{ji}(t)$；

(3) $\lim_{t\to 0} \Phi(x,t) = \begin{cases} 0, & x > 0, \\ 1, & x < 0; \end{cases}$

(4) $\dfrac{\Phi(x_1,t)}{\Phi(x_2,t)} = \Phi(x_1 - x_2, t)$；

(5) $\forall t > 0$，都有 $\Phi(0,t) = 1$，

则马氏链的平稳分布为

$$v_i(t) = \frac{\Phi(f(i),t)}{\sum_{j\in D} \Phi(f(j),t)}, \quad \forall i \in D, \tag{3.19}$$

且

$$\lim_{t\to 0} v_i(t) = \begin{cases} \dfrac{1}{|D_{\mathrm{OPT}}|}, & i \in D_{\mathrm{OPT}}, \\ 0, & i \notin D_{\mathrm{OPT}}. \end{cases} \tag{3.20}$$

证明 定理 3.8 和定理 3.9 已保证平稳分布存在且唯一. 下面仅证明(3.19)为平稳分布. 由(1)知(3.19)定义的值为正数. 又

由(2)可知下列等式成立.

$$\begin{aligned}\sum_j v_j(t)p_{ji}(t) &= \sum_{j\neq i}\frac{\Phi(f(j),t)}{K}G_{ji}(t)A_{ji}(t)+v_i(t)p_{ii}(t)\\ &= \frac{\Phi(f(i),t)}{K}\sum_{j\neq i}G_{ij}(t)A_{ij}(t)+v_i(t)p_{ii}(t)\\ &= v_i(t)\Big[\sum_{j\neq i}p_{ij}(t)+p_{ii}(t)\Big]\\ &= v_i(t),\end{aligned}$$

其中,$K=\sum_{j\in D}\Phi(f(j),t)$. 由定理3.5平稳分布和唯一性知(3.19)定义的值为平稳分布. 再由定理中的(3)(4)(5)知

$$\lim_{t\to 0}v_i(t)=\lim_{t\to 0}\frac{\Phi(f(i),t)}{\sum_j\Phi(f(j),t)}=\lim_{t\to 0}\frac{\dfrac{\Phi(f(i),t)}{\Phi(f_{\mathrm{OPT}},t)}}{\dfrac{\sum_j\Phi(f(j),t)}{\Phi(f_{\mathrm{OPT}},t)}}$$

$$=\lim_{t\to 0}\frac{\Phi(f(i)-f_{\mathrm{OPT}},t)}{\sum_j\Phi(f(j)-f_{\mathrm{OPT}},t)}=\begin{cases}\dfrac{1}{|D_{\mathrm{OPT}}|}, & i\in D_{\mathrm{OPT}},\\ 0, & i\notin D_{\mathrm{OPT}},\end{cases}$$

其中,f_{OPT}表示最优目标值. (3.20)得证. □

现在讨论(3.6)在一些特定的条件下,平稳分布的存在性和表达式形式.

定理3.13　若$A_{ij}(t)$,$G_{ij}(t)$满足定理3.10中除(2)以外的条件,且定理3.10条件(2)改为:任何两个状态i和j或相互为邻居或不为,且$G_{ij}(t)$满足

$$\forall\, i\in D,\quad G_{ij}=\begin{cases}\dfrac{1}{|N(i)|}, & j\in N(i),\\ 0, & j\notin N(i),\end{cases}$$

状态空间D对邻域连通. 则平稳分布是

$$v_i(t)=\frac{|N(i)|A_{i_0 i}(t)}{\sum_{j\in D}|N(j)|A_{i_0 j}(t)},\quad \forall\, i\in D. \tag{3.21}$$

证明 由状态空间对邻域连通的条件、定理 3.10 的(4) 和(5),得到定理 3.8 的条件满足,因此马氏链为不可约的.很容易验证定理 3.9 的条件满足,所以非周期性得证.由定理 3.5 得到分布存在且唯一.下面验证(3.21) 是平稳分布.

$$\sum_j v_j(t) p_{ji}(t)$$

$$= \sum_{j\neq i, i\in N(j)} \frac{|N(j)|}{K} A_{i_0j}(t) G_{ji} A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0i}(t)$$

$$\times \left(1 - \sum_{l=1, l\neq i}^{|D|} G_{il}(t) A_{il}(t)\right) + \sum_{j: i\notin N(j)} \frac{|N(j)|}{K} A_{i_0j}(t) G_{ji} A_{ji}(t)$$

$$= \sum_{j\neq i: i\in N(j)} \frac{1}{K} A_{i_0j}(t) A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0i}(t)$$

$$\times \left(1 - \sum_{l\in N(i), l\neq i} G_{il}(t) A_{il}(t)\right)$$

$$= \sum_{j\neq i: i\in N(j)} \frac{1}{K} A_{i_0j}(t) A_{ji}(t) + \frac{|N(i)|}{K} A_{i_0i}(t) - \sum_{l\in N(i), l\neq i} \frac{1}{K} A_{i_0i} A_{il}(t)$$

其中,$K = \sum_{j\in D} |N(j)| A_{i_0j}(t)$.由任何两个状态或相互为邻居或不为,考察

$$\sum_{j\neq i: i\in N(j)} \frac{1}{K} A_{i_0j}(t) A_{ji}(t) - \sum_{l\in N(i), l\neq i} \frac{1}{K} A_{i_0i} A_{il}(t)$$

$$= \sum_{j\neq i: i\in N(j), f(i)\leqslant f(j)} \frac{1}{K} A_{i_0j}(t) + \sum_{j\neq i: i\in N(j), f(i)> f(j)} \frac{1}{K} A_{i_0i}(t)$$

$$- \sum_{l: l\in N(i), l\neq i, f(l)<f(i)} \frac{1}{K} A_{i_0i} - \sum_{l: l\in N(i), l\neq i, f(l)\geqslant f(i)} \frac{1}{K} A_{i_0l}$$

$$= 0$$

因此,(3.21) 是平稳分布. □

例 3.3 四个状态 1,2,3,4 的邻居分别为

$$N(1) = \{1,2\},\ N(2) = \{1,2,3,4\},$$

$$N(3) = \{2,3,4\},\ N(4) = \{2,3,4\}.$$

很容易验证,它们满足或互为邻居或互不为邻居的假设.但需要注

意的是定理 3.13 定义的 $G_{ij}(t)$ 不满足 $G_{ij}=G_{ji}$，如 $G_{12}=\frac{1}{2}$ 和 $G_{21}=\frac{1}{4}$. 这个例子说明定理 3.13 成立的条件比定理 3.10 成立的条件要广泛. □

满足 $A(t)$ 和 $G(t)$ 为非周期、不可约还有其他形式的一步转移概率，如文献[3]给出的发生概率为

$$\text{存在 } |D|\times|D| \text{ 矩阵 } Q \text{ 满足}$$
$$\forall\, i,j\in D \text{ 使得 } Q_{ij}=Q_{ji};$$
$$\forall\, i\in D, j\in N(i) \text{ 使得 } G_{ij}=\frac{Q_{ij}}{\sum\limits_{l\in D}Q_{il}},$$

它的平稳分布为：

$$v_i(t)=\frac{\left(\sum\limits_{l}Q_{il}\right)A_{i_0i}(t)}{\sum\limits_{j\in D}\left(\sum\limits_{l}Q_{jl}\right)A_{i_0j}(t)},\quad \forall\, i\in D. \tag{3.22}$$

(3.22) 的证明同定理 3.13 的证明类似，读者可以自己证明.

文献[4]给出接受概率为

$$A_{ij}(t)=\left(1+\exp\left(-\frac{f(j)-f(i)}{t}\right)\right)^{-1}. \tag{3.23}$$

它不满足定理 3.10 的(2)(3)，但它的平稳分布是

$$v_i(t)=\frac{\exp\left(-\frac{f(i)-f_{\mathrm{OPT}}}{t}\right)}{\sum\limits_{j\in D}\exp\left(-\frac{f(j)-f_{\mathrm{OPT}}}{t}\right)},\quad \forall\, i\in D. \tag{3.24}$$

3.4　非时齐算法收敛性简介

非时齐算法一步转移概率的形式为：

$$p_{ij}(k-1,k)=\Pr(X(k)=j\,|\,X(k-1)=i))$$

$$=\begin{cases} G_{ij}(t_k)A_{ij}(t_k), & \forall j \neq i, \\ 1-\sum\limits_{l=\cdot,l\neq i}^{|D|} G_{il}(t_k)A_{il}(t_k), & j=i, \end{cases} \tag{3.25}$$

其中,$t_{k-1} \geqslant t_k$, $\lim\limits_{k\to\infty} t_k = 0$. 一步转移概率矩阵为:$\boldsymbol{P}(k-1,k) = (p_{ij}(k-1,k))_{|D|\times|D|}$. 多步转移概率 $p_{ij}(m,k)$ 表示第 m 步在状态 i,第 k 步在状态 j 的转移概率.

出于第 3.2 节对非时齐算法的要求,我们需要了解收敛性的一些定义.

定义 3.1 若非时齐马氏链满足下列条件,则称为弱遍历(weakly ergodic):

$$\forall\, i,j,l \in D,\ m \geqslant 1,$$
$$\lim_{k\to\infty}(p_{il}(m,k) - p_{jl}(m,k)) = 0.$$

这个定义说明,无论起点如何,当运动步数增加时,到达同一状态的概率渐近相同. 这是一种失去记忆功能.

定义 3.2 若存在向量 $\boldsymbol{v} = (v_1, v_2, \cdots, v_{|D|})$,满足

$$\sum_{i=1}^{|D|} v_i = 1, \quad \forall\, i, v_i \geqslant 0,$$

且有

$$\forall\, i,j \in D,\ m \geqslant 1,$$
$$\lim_{k\to\infty} p_{ij}(m,k) = v_j,$$

则称非时齐马氏链为强遍历(strongly ergodic).

定义 3.2 的意义是马氏链的渐近稳定性. 与时齐算法相同的思路,需要研究(3.25) 的一些限定条件,使得非时齐马氏链具有定义 3.2 的强遍历性质. 同时,使得分布 $\boldsymbol{v} = (v_1, v_2, \cdots, v_{|D|})$ 具有全局最优的性质. 我们不加证明地给出下面的定理.

定理 3.14[5] 一个非时齐的马氏链是弱遍历的充分必要条件是存在一个严格上升的正序列 $\{k_l\}_{l=0,1,\cdots}$,使得

$$\sum_{l=0}^{\infty}(1-\tau_1(\boldsymbol{P}(k_l,k_{l+1})))=\infty, \tag{3.26}$$

其中，$\tau_1(\boldsymbol{P})=1-\min\limits_{i,j}\sum\limits_{l=1}^{n}\min(p_{il},p_{jl})$，$\boldsymbol{P}=(p_{ij})_{n\times n}$.

定理 3.15[6]　一个非时齐的马氏链是强遍历的充分必要条件是：马氏链是弱遍历且存在 $\boldsymbol{P}(k-1,k)$ 的特征值为 1 的特征向量 $\boldsymbol{V}(k)=(v_1(k),v_2(k),\cdots,v_{|D|}(k))$，使得 $\sum\limits_{i=1}^{|D|}|v_i(k)|=1$，且

$$\sum_{k=0}^{\infty}\sum_{i=1}^{|D|}|v_i(k)-v_i(k+1)|<\infty, \tag{3.27}$$

另外，若 $\boldsymbol{V}=\lim\limits_{k\to\infty}\boldsymbol{V}(k)$，则 $\boldsymbol{V}$ 满足 $\lim\limits_{k\to\infty}p_{ij}(m,k)=v_j$.

分析定理 3.14 和定理 3.15，在给定温度 t_k 以后和一步转移概率满足一定的条件时，由 3.3 节对时齐算法的讨论，存在平稳分布 $\boldsymbol{V}(t_k)=(v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k))$. 现在的主要内容是讨论如何选择 t_k，使得：

(1) 马氏链满足定理 3.14 的(3.26)，

(2) $\boldsymbol{V}(t_k)=(v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k))$ 满足(3.27)，

(3) 极限分布 V 满足全局收敛 $\lim\limits_{k\to\infty}\Pr\{X(k)\in D_{\mathrm{OPT}}\}=1$.

定理 3.16　在温度 t 时，一步转移概率(3.5)中的 $A(t)$ 按(3.7)定义，即

$$A_{ij}(t)=\begin{cases}1, & f(i)\geqslant f(j),\\ \exp\left(-\dfrac{\Delta f_{ij}}{t}\right), & f(i)<f(j),\end{cases}$$

$G(t)$ 为

$$\forall\, i\in D, G_{ij}=\begin{cases}\dfrac{1}{|D|}, & j\in N(i),\\ 0, & j\notin N(i),\end{cases} \tag{3.28}$$

当

$$\text{存在 } k_0\geqslant 2\text{，使得 }\forall\, k\geqslant k_0,\quad t_k\geqslant\frac{|D|\Delta f_{\max}}{\log k}, \tag{3.29}$$

其中,

$$\Delta f_{\max} = \max\{f(i) \mid i \in D\} - \min\{f(i) \mid i \in D\}, \quad (3.30)$$

则马氏链 $\{X(t_k)\}_{k=1,2,\cdots}$ 为强遍历.

文献[7]首先证明了满足定理3.16条件的马氏链具有弱收敛性. 再由定理 3.10,(3.7) 和(3.28) 决定的马氏链有(3.24) 的平稳分布,即

$$v_i(t_k) = \frac{\exp\left(-\frac{f(i)-f_{\text{OPT}}}{t_k}\right)}{\sum_{j\in D}\exp\left(-\frac{f(j)-f_{\text{OPT}}}{t_k}\right)}, \quad \forall\, i \in D. \quad (3.31)$$

可以验证(3.31) 满足(3.27)(留作练习),于是,非时齐马氏链是强遍历. 由定理 3.11 和 $A(t)$ 的性质,知本定理对应的非时齐马氏链收敛到全局最优解,即

$$\lim_{k\to\infty} P(X(k) \in D_{\text{OPT}}) = 1.$$

另外,温度下降的最快速度为

$$t_k = \frac{|D|\Delta f_{\max}}{\log k}. \quad (3.32)$$

关于非时齐算法的下降速度还有一些其他精细估计. 在定理3.16 的条件下,文献[8]中证明了:温度下降满足

$$\forall\, k \geqslant 2, \quad t_k \geqslant \frac{n\Delta}{\log k} \quad (3.33)$$

其中,$\Delta = \max\limits_{i,j}\{f(j) - f(i) \mid i \in D, j \in N(i)\}$,$n$ 是从任何一个状态可达到最优解的转移步数的上限,明显不超过 $|D|$,k 是 n 的倍数.

非时齐马氏链收敛的一个充分必要性在文献[9]中给出,下面给出谷深的定义.

定义 3.3 若存在 $i = l_0, l_1, \cdots, l_p = j$ 使得

$$G_{l_k l_{k+1}} > 0, \quad k = 0,1,\cdots,p-1,$$

$$f(l_k) \leqslant L, \quad k = 0,1,\cdots,p,$$

其中，$G(t)$ 为(3.6)的产生概率，则称状态 i 以高度 L 达 j. 一个深度不超过 L 的谷定义为 V_L，满足

$$V_L = \{j \in D \mid \forall\, i \in V_L, i \text{ 以高度 } L \text{ 达 } j\}.$$

谷的实际深度可以由

$$\underline{V_L} = \min\{f(j) \mid j \in V_L\},$$

$$\overline{V_L} = \min\{f(j) \mid j \notin V_L \wedge \exists\, i \in V_L, G_{ij} > 0\},$$

的差决定，即谷深 $D = \overline{V_L} - \underline{V_L}$. 谷中的一个局部最小点 i 可以看成状态 i 无法在高度 $f(i)$ 达到其他状态 j.

定理 3.17　若一步转移概率由(3.25)的形式给出，且 $A(t_k)$ 取(3.7)形式，$G(t_k)$ 满足

(1) $G(t_k)$ 决定的马氏链对任意给出的 i 和 j，存在 $i = l_0, l_1, \cdots, l_p = j$，使得

$$G_{l_k l_{k+1}} > 0, \quad k = 0,1,\cdots,p-1;$$

(2) 对一实数 L 和任意两个状态 i 和 j，i 以高度 L 可达 j 的充分必要条件是 j 以高度 L 可达 i；

(3) $t_k \geqslant t_{k+1}, \lim\limits_{k\to\infty} t_k = 0$；若 d 是所有不包含最优解谷的最大谷深，则有

$$\lim_{k\to\infty} \Pr(X(k) \in D_{\text{OPT}}) = 1$$

的充分必要条件是

$$\sum_{k=1}^{\infty} \exp\left(-\frac{d}{t_k}\right) = \infty. \tag{3.34}$$

从定理 3.16 和定理 3.17 可以看出，模拟退火非时齐算法全局收敛的一个充分条件是温度下降具有形式

$$t_k = \frac{\Gamma}{\log k}, \quad k > 1, \tag{3.35}$$

其中 Γ 是一个同问题相关的参数. 如定理 3.16 中，

$$\Gamma \geqslant |D| \Delta f_{\max},$$

(3.33)式中 $\Gamma \geqslant n\Delta$,定理3.17中的 Γ 取所有不包含最优解谷的最大深度.

3.5 实现的技术问题

前面各节介绍了模拟退火的算法结构、模型,时齐和非时齐算法的收敛理论.本节从应用的角度讨论算法实现中的一些技术问题.主要包括解的形式、邻域的构成、初始温度的选取、温度下降的规则、每一温度马氏链的迭代步长和停止规则等.由非时齐算法的收敛理论可知,其下降的规则是有固定形式的,因此,本节所讨论的技术问题以时齐算法为主.

从前几节的理论研究还可以看出,按理论要求达到平稳分布来应用模拟退火算法是不可能的,这是因为时齐的算法要在每一个温度迭代无穷步以达到平稳分布,而非时齐要求温度下降的迭代步数是指数次的.从应用的角度来看,在可接受的时间里得到满意的解就可以了,因此本节介绍的技术问题无法保证模拟退火算法得到全局最优解.应用这些技术的模拟退火算法还是一种启发式算法.

由于本节是讨论应用的技术问题,因此,它同理论部分有一定的联系,同时可能相差较远.部分技术问题虽说给出了一些理论结果,但这些结果是指导性的.

3.5.1 解的形式和邻域结构

解的形式和邻域结构是一个老话题,在第2章中已讨论.本节再度讨论这个问题以引起读者的注意.解的表现形式直接决定于邻域的构造.

第1个问题如 $f(x)=x^2, 0\leqslant x\leqslant 100$,$x$ 为整数.它的解可以采用0-1编码,即

$$S=\begin{pmatrix}1 & 2 & \cdots & 7\\ * & * & \cdots & *\end{pmatrix}.$$

其中,解 S 的上面一行表示 7 个位置,下面的 * 是 0-1 码,对应 x 的二进制码,解的一个邻域自然可以构造成

$$N(S')=\{S\,|\,S\text{ 是一个 0-1 码且 }|S-S'|=\sum_{i=1}^{7}|s_i-s_i'|\leqslant k,k\geqslant 1\text{ 整数}\}.\tag{3.36}$$

第 2 个问题如第 1 章例 1.2 的 TSP 这一类问题,可采用 n 个城市的一个排序表示问题的一个解.很直观可通过城市间不同位置交换构造邻域.以上两个问题的共同特征是:(1) 每一个邻域所含的状态个数相同,如第 1 个问题中每个邻域中都有 $C_7^k+C_7^{k-1}+\cdots+C_7^0$ 个邻居,TSP 中若采用 2-opt 则每个邻域有 C_n^2+1 个邻居;(2) 每个邻居都是可行解;(3) 空间中的任何两个状态可达.于是由时齐算法的理论,只要接受概率满足一定的条件,一定以概率 1 收敛到全局最优解.

同样解的表达式和邻域结构,将第 1 个问题的方法应用到背包问题时,第 2 个问题的方法应用到车间作业问题(第 2 章 2.4.2 节).由于背包问题有包容积约束的限制和车间作业的死锁现象,无法保证每个邻域中邻居都是可行解.处理这个问题有两类方法:一个是用罚函数将不可行解可行化,这使得问题转化为上面讨论的两个问题,原有的方法可以继续使用.采用罚函数的方法对不可行解进行处理时,一个主要缺陷是引起搜索区域的扩大,从而使计算时间增加.为了克服这个不足,另一个方法是研究解和邻域结构,从理论上保证模拟退火算法以概率 1 收敛到全局最优解.这一个方法要求对问题有相当深入的了解,因此,可以说其难度是比较大的.通过下面的例子来了解罚函数引起了搜索区域的扩大.

例 3.4　背包问题:三个物品,体积分别为{5,3,1},包的容积为 4,则{(0,0,0),(0,1,0),(0,0,1),(0,1,1)}是可行解集合.

若采用罚值的方法，则解{(1,0,0), (1,1,0), (1,0,1), (1,1,1)}也成了搜索的对象，于是搜索空间扩大了一倍. □

罚值的方法可以很简单地保证理论要求的不可约和非周期性. 但如此计算一定会增加计算时间，而有时增加时间之多是不可以接受的. 如按例 1.2 的 TSP 的表示方法，若商人走(i,j)弧则$x_{ij}=1$，否则$x_{ij}=0$. 这样每一个解的维数是$n\times(n-1)$，每个分量取 0 或 1，一共有$2^{n(n-1)}$个解. 但是这些解只有一部分是可行解. 若假设每两个节点有弧相连，固定一个城市为始终点，则对应所有可行解可以用所有城市的排列表示，其排列最大数，即可行解数为$(n-1)!$. 若采用例 1.2 解的表达形式和对不可行解采用罚值的方法，则将搜索$2^{n(n-1)}-(n-1)!$个不可行解. 搜索空间扩大$\frac{2^{n(n-1)}}{(n-1)!}$倍. 当$n=5$时，扩大倍数$\frac{2^{n(n-1)}}{(n-1)!}\approx 4.37\times 10^4$，当$n=10$时，扩大倍数$\frac{2^{n(n-1)}}{(n-1)!}\approx 3.41\times 10^{21}$.

为了节省时间，可以考虑在可行解集合内构造邻域，如第 2 章车间作业问题的邻域构造. 后一种方法是需要对问题本身的结构有很好的了解且需要较强的理论支持. 这种方法需要类似 3.4 节和 3.5 节那样从理论上证明对应的马氏链有全局最优性.

在此，邻域的构造同计算时间紧密联系在一起，也就产生了对实际问题的理论分析同实际应用的矛盾. 从科学的角度，我们希望对问题进行较深入的理论研究，构造比较精巧的邻域结构，这样可以节省大量的计算时间. 不得不承认，在大量的实际应用中，受各种客观因素的影响，人们对问题还无法深入地了解，且在有些情况下，人们关注的是应用的效果，而不去研究问题的结构和收敛性，直接将算法套用于实际问题.

3.5.2 温度参数的控制

温度参数是模拟退火时齐算法中一个最关键的参数之一. 主

要包括起始温度的选取、温度的下降方法和停止温度的确定等.

1. 起始温度的选取

从理论上来说,起始温度 t_0 应保证平稳分布中每一状态的概率相等,也就是使

$$\exp\left(-\frac{\Delta f_{ij}}{t_0}\right)\approx 1,$$

很容易估计一个值为

$$t_0 = K\delta,\ K\ 充分大的数, \tag{3.37}$$

其中,

$$\delta = \max\{f(j)\mid j\in D\} - \min\{f(j)\mid j\in D\}.$$

实际计算中,可以选 $K = 10,20,100\cdots$ 等试验值.

对一些问题,有时可以简单地估计 δ. 如例 1.2 的 TSP,记对应的距离矩阵为 $(d_{ij})_{n\times n}$,则

$$\max\{f(j)\mid j\in D\}\leqslant \sum_{i=1}^{n}\max\{d_{ij}\mid j\neq i,j=1,2,\cdots,n\},$$

$$\min\{f(j)\mid j\in D\}\geqslant \sum_{i=1}^{n}\min\{d_{ij}\mid j\neq i,j=1,2,\cdots,n\},$$

$$\delta_1=\sum_{i=1}^{n}\max\{d_{ij}\mid j\neq i,j=1,2,\cdots,n\} - \sum_{i=1}^{n}\min\{d_{ij}\mid j\neq i,j=1,2,\cdots,n\}.$$

则可用 δ_1 替代(3.37) 中的 δ.

再如 2.4.2 的车间作业调度问题,记第 $i(1\leqslant i\leqslant n)$ 个工件的第 $j(1\leqslant j\leqslant n_i)$ 工序的加工时间为 p_{ij}. 完成的最短时间是所有工件都从时刻零开始加工且每一台机器都没有闲置,此时

$$\min\{f(j)\mid j\in D\}\geqslant \max\left\{\sum_{j=1}^{n_i}p_{ij}\mid i=1,2,\cdots,n\right\}.$$

完成的最长时间是调度最坏情况

$$\max\{f(j)\mid j\in D\}\leqslant \sum_{i=1}^{n}\sum_{j=1}^{n_i}p_{ij}.$$

令

$$\delta_2 = \sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ij} - \max\left\{\sum_{j=1}^{n_i} p_{ij} \mid i = 1,2,\cdots,n\right\}.$$

则可用 δ_2 替代(3.37)中的 δ.

但有的时候,会出现 δ 比较难估计或很难精确地知道最大值和最小值而使上面的估计太粗.另外,K 的选取过大会造成计算时间的增加,K 过小则使算法过早陷入局部最优点.由此产生数值计算估计和统计推断估计方法.

数值计算估计方法的基本思想是给出一个值 χ_0(χ_0 接近 1,如 $\chi_0 = 0.9, 0.8$ 等),对给定的初始温度 t_0,用以下的算法:

初始温度数值计算算法

STEP1 给定一个常量 T;初始温度 t_0;χ_0;$R_0 = 0$;$k := 1$;

STEP2 在这个温度迭代 L 步(L 为一个给定的常数),分别记录模拟退火算法中接受和被拒绝的状态的个数,计算接受的状态数同迭代步数 L 的比率 R_k;

STEP3 当 $|R_k - \chi_0| < \epsilon$ 时,停止计算;否则,当 R_{k-1} 和 $R_k < \chi_0$ 时,则 $k := k+1, t_0 := t_0 + T$,返回 STEP2;当 R_{k-1} 和 $R_k \geqslant \chi_0$ 时,则 $k := k+1, t_0 := t_0 - T$,返回 STEP2;当 $R_{k-1} \geqslant \chi_0$ 且 $R_k \leqslant \chi_0$ 时,则 $k := k+1, t_0 := t_0 + T/2$, $T := T/2$,返回 STEP2;当 $R_{k-1} \leqslant \chi_0$ 且 $R_k \geqslant \chi_0$ 时,则 $k := k+1, t_0 := t_0 - T/2$,返回 STEP2.

通过数值计算,可以估计出温度 t_0.

针对(3.37)中 $\delta = \max\{f(j) \mid j \in D\} - \min\{f(j) \mid j \in D\}$ 比较难得到,通常采用统计的方法估计费用函数的上下限.统计的方法总是在一定的概率分布下研究问题的均值、方差等.假设 $\{f(i) \mid i \in D\}$ 是一个大样本空间,且服从正态分布,即 $\{f(i) \mid i \in$

$D\}$ 的密度函数为

$$g(x)=\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

从状态空间 D 中随机选 n 个独立样本 $\{X_i|i=1,2,\cdots,n\}$，样本均值统计量为

$$\overline{f(X,t)}=\frac{1}{n}\sum_{i=1}^{n}f(X_i), \tag{3.38}$$

样本方差统计量为

$$f^2(X,t)=\frac{1}{n-1}\sum_{i=1}^{n}(f(X_i)-\overline{f(X,t)})^2. \tag{3.39}$$

由概率论的分布特性，(3.38)，(3.39) 分别是 μ,σ^2 的统计量. 用它们作为近似，以 3σ 原则，目标值以 99.97% 落入 $(\mu-3\sigma,\mu+3\sigma)$（参考[10]），简单的区间近似为

$$\left(\overline{f(X,t)}-3\sqrt{f^2(X,t)},\overline{f(X,t)}+3\sqrt{f^2(X,t)}\right).$$

估计(3.37)的

$$\delta=6\sqrt{f^2(X,t)}.$$

2. 时齐算法的温度下降方法

由于非时齐算法温度下降具有固定的形式（见 3.4 节），因此，这里只讨论时齐算法的温度下降方法. 从实际应用来看，非时齐算法的下降速度太慢，可以有变通的形式以利于节省算法时间，在此不讨论.

时齐算法的理论要求温度下降到零，整个系统以概率 1 收敛全局最优解. 无论直观理解还是理论要求，温度总是下降的，因此，一个非常直观的下降方法是：

(1) $t_{k+1}=\alpha t_k,k\geqslant 0$，其中 $0<\alpha<1$. α 越接近 1 温度下降得越慢. 这种方法简单易行，很受应用人员的欢迎. 它的每一步以相同的比率下降.

(2) $t_k=\dfrac{K-k}{K}t_0$，其中 t_0 为起始温度，K 为算法温度下降的

总次数.这一下降的方法的优点是易于操作,而且可以简单地控制温度下降的总步数.它的每一步下降长度相等.

(3) 这一下降方法以理论中的平稳分布为依据,通过推导而得到.为了方便以后的讨论,首先在温度 t,定义

$$\langle f(t)\rangle = \sum_{i\in D} f(i)v_i(t), \tag{3.40}$$

$$\langle f^2(t)\rangle = \sum_{i\in D} f^2(i)v_i(t), \tag{3.41}$$

$$\overline{f(t)} = \frac{1}{n}\sum_{i=1}^{n} f(X_i), \tag{3.42}$$

$$\overline{f^2(t)} = \frac{1}{n}\sum_{i=1}^{n} f^2(X_i). \tag{3.43}$$

(3.40) 和(3.41) 分别是平稳分布时的期望值和二阶矩值.(3.42) 和(3.43) 为温度 t 模拟退火 n 个样本的均值和二阶矩值.为以后使用方便,先讨论(3.40) 和(3.41) 的一些基本性质.

定理3.18 若 $A(t)$ 和 $G(t)$ 满足定理3.13的条件,且 $A(t)$ 和 $G(t)$ 分别满足(3.6) 和(3.7),则

(1) $\dfrac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t} = \dfrac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t^2}$,

(2) $\langle f(t)\rangle$ 为变量 t 的单调升函数,且 $\langle f(t)\rangle \to f(i_{\mathrm{OPT}})$, $t\to 0$.

(3) 当 $i\neq i_{\mathrm{OPT}}$ 时,平稳分布 $v_i(t)$ 在 $t=0$ 的一个邻域内为 t 的单调升函数.当 $i=i_{\mathrm{OPT}}$ 时,平稳分布 $v_i(t)$ 为 t 的单调减函数.

证明 (1) $\dfrac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t} = \sum_{i\in D} f(i)\dfrac{\mathrm{d}v_i(t)}{\mathrm{d}t}$.

由

$$v_i(t)=\frac{|N(i)|\exp\left(-\dfrac{f(i)-f_{\mathrm{OPT}}}{t}\right)}{\sum_{j\in D}|N(i)|\exp\left(-\dfrac{f(j)-f_{\mathrm{OPT}}}{t}\right)}$$

$$= \frac{|N(i)|\exp\left(-\frac{f(i)}{t}\right)}{\sum_{j\in D}|N(i)|\exp\left(-\frac{f(j)}{t}\right)}, \quad \forall\, i \in D,$$

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t} = \frac{\mathrm{d}\left[\frac{|N(i)|\exp\left(-\frac{f(i)}{t}\right)}{\sum_{j\in D}|N(j)|\exp\left(-\frac{f(j)}{t}\right)}\right]}{\mathrm{d}t}$$

$$= \frac{|N(i)|\exp\left(-\frac{f(i)}{t}\right)f(i)}{t^2K}$$

$$- \frac{|N(i)|\exp\left(-\frac{f(i)}{t}\right)\sum_{j\in D}|N(j)|\frac{f(j)}{t^2}\exp\left(-\frac{f(j)}{t}\right)}{K^2}$$

$$= \frac{v_i(t)f(i)}{t^2} - \frac{v_i(t)\langle f(t)\rangle}{t^2}.$$

其中，$K = \sum_{j\in D}|N(j)|\exp\left(-\frac{f(j)}{t}\right)$. 于是，可得

$$\frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t} = \frac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t^2}.$$

(2) 由

$$\frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t} = \frac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t^2}$$

$$= \frac{\sum_{i\in D}(f(i) - \langle f(i)\rangle)^2 v_i(t)}{t^2}$$

$$\geqslant 0,$$

得 $\langle f(t)\rangle$ 对 t 单调上升. 再由定理 3.11 很易证 $\langle f(t)\rangle \to f(i_{\mathrm{OPT}})$，$t \to 0$.

(3) 由(1)的证明，

$$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t}=\frac{v_i(t)f(i)}{t^2}-\frac{v_i(t)\langle f(t)\rangle}{t^2}.$$

当 $i\neq i_{\mathrm{OPT}}$ 时，若 $f(i)-f(i_{\mathrm{OPT}})=\epsilon>0$，由(2)的结论，存在 $\delta>0$，当 $0\leqslant t<\delta$ 时，$\langle f(t)\rangle<f(i_{\mathrm{OPT}})+\epsilon$. 于是，$\frac{\mathrm{d}v_i(t)}{\mathrm{d}t}>0$，故 $v_i(t)$ 在 $t=0$ 的一个邻域内为 t 的单调升函数. 同法可证，当 $i=i_{\mathrm{OPT}}$ 时，平稳分布 $v_i(t)$ 为 t 的单调减函数. □

假设在相邻温度，其平稳分布的变化相对稳定. 由定理 3.18，平稳分布的变化相对稳定等价于用数学公式表达：给定一个较小的正数 δ，相邻温度的平稳分布应满足

$$\forall i\in D,\ \frac{1}{1+\delta}<\frac{v_i(t_k)}{v_i(t_{k-1})}<1+\delta,\ k=0,1,\cdots.\tag{3.44}$$

按定理 3.13 且 $A(t)$ 满足(3.7)，(3.44) 近似为

$$\exp\left(-\frac{f(i)-f(i_{\mathrm{OPT}})}{t_k}\right)<(1+\delta)\exp\left(-\frac{f(i)-f(i_{\mathrm{OPT}})}{t_{k+1}}\right).\tag{3.45}$$

(3.45) 变形为

$$\frac{f(i)-f(i_{\mathrm{OPT}})}{t_{k+1}}<\log(1+\delta)+\frac{f(i)-f(i_{\mathrm{OPT}})}{t_k}.\tag{3.46}$$

进一步可得

$$t_{k+1}\left(1+\frac{t_k\log(1+\delta)}{f(i)-f(i_{\mathrm{OPT}})}\right)>t_k.\tag{3.47}$$

假设在温度 t_k，$\{f(i)-f(i_{\mathrm{OPT}})\mid i=1,2,\cdots,|D|\}$ 出现的频率服从分布 $\{v_1(t_k),v_2(t_k),\cdots,v_{|D|}(t_k)\}$，其对应的均值和方差分别为 $\langle f(t_k)\rangle$，$\langle f^2(t_k)\rangle$，其随机抽样均值的统计量为

$$u(t_k)=\overline{f(t_k)}-f(i_{\mathrm{OPT}}),$$

方差的统计量为

$$\sigma^2(t_k)=\overline{f^2(t_k)}-\left(\overline{f(t_k)}\right)^2.$$

则以概率 99.74% 费用 $\{f(i)-f(i_{\mathrm{OPT}})\mid i=1,2,\cdots,|D|\}$ 落在

区间

$$(-3\langle f^2(t_k)\rangle + \langle f(t_k)\rangle, \langle f(t_k)\rangle + 3\langle f^2(t_k)\rangle).$$

近似为

$$(-3\sigma(t_k) + \mu(t_k), \mu(t_k) + 3\sigma(t_k)).$$

由上面的近似结果，当

$$t_{k+1}\left(1 + \frac{t_k \log(1+\delta)}{\mu(t_k) + 3\sigma(t_k)}\right) > t_k$$

时，(3.47) 同样成立. 于是温度的变化

$$t_{k+1} = t_k\left(1 + \frac{t_k \log(1+\delta)}{\mu(t_k) + 3\sigma(t_k)}\right)^{-1}. \tag{3.48}$$

计算 $\mu(t_k)$, $\sigma(t_k)$ 时，要求对每一个温度 t_k 采用累计求和的方法，而不需要记录所有的状态.

Lundy 和 Mees[11] 采用基本相同的想法，使(3.44) 近似为

$$\forall\, i \in D, \text{使得 } \exp\left(\frac{(f(i) - f(i_{\mathrm{OPT}}))(t_k - t_{k+1})}{t_k t_{k+1}}\right) < (1+\delta),$$

当 U 是一个 $f(i) - f(i_{\mathrm{OPT}})$ 的上界，r 是一个充分小的正数时，下式可以保证上式的成立.

$$\frac{t_k - t_{k+1}}{t_k t_{k+1}} = \frac{r}{U},$$

于是可以得降温规则

$$t_{k+1} = t_k\left(1 + \frac{r t_k}{U}\right)^{-1}. \tag{3.49}$$

文献[12] 基于 $A(t)$ 是(3.7) 的形式，得(练习题)

$$\frac{\mathrm{d}}{\mathrm{d}\log t}\langle f(t)\rangle = \frac{\langle f^2(t)\rangle - \langle f(t)\rangle^2}{t},$$

近似为

$$\frac{\overline{f(t_{k+1})} - \overline{f(t_k)}}{\log t_{k+1} - \log t_k} \approx \frac{\overline{f^2(t_k)} - (\overline{f(t_k)})^2}{t_k},$$

由此推导

$$t_{k+1} = t_k \exp\left[\frac{t_k\left(\overline{f(t_{k+1})} - \overline{f(t_k)}\right)}{\overline{f^2(t_k)} - \left(\overline{f(t_k)}\right)^2}\right].$$

若要求 $\overline{f(t_{k+1})} - \overline{f(t_k)}$ 不超过它的标准差，即

$$\overline{f(t_{k+1})} - \overline{f(t_k)} = -\lambda\sqrt{\overline{f^2(t_{k+1})} - \left(\overline{f(t_k)}\right)^2},\ 0 < \lambda < 1,$$

则有文献[12]中的结果

$$t_{k-1} = t_k \exp\left[-\frac{t_k\lambda}{\sqrt{\overline{f^2(t_k)} - \left(\overline{f(t_k)}\right)^2}}\right]. \tag{3.50}$$

3. 时齐算法每一温度的迭代长度规则

同样，实际计算中按理论达到平稳分布是不可能的，只能近似这一结果. 下面就常用的方法给予讨论.

(1) 固定长度

这是一个简单的方法，在每一温度，迭代相同的步数，步数的选取同问题的大小有关，通常采用与邻域大小直接相关的规则. 如 TSP 的两城市位置交换所定义的邻域，它的大小是 $\frac{n(n-1)}{2}$（其中 n 是城市数），在同一温度，它的迭代步长可取 $n, \frac{n}{2}, n^2, \frac{n^2}{2}, n^k (k \geqslant 3)$ 等.

(2) 由接受和拒绝的比率来控制迭代步数

当温度很高时，每一个状态被接受的频率基本相同，而且几乎所有的状态都被接受. 此时，在同一温度应使迭代的步数尽量小. 当温度渐渐变低时，越来越多的状态被拒绝. 如果在此温度的迭代太少，则可能造成过早地陷入局部最优状态. 比较直观和有效的方法是随着温度的下降，将同一温度的迭代步长增加. 实现的一种方法是给定一个充分大的步长上限 U 和一个接受次数指标 R，当接受次数等于 R 时，在此温度不再迭代而使温度下降，否则，一直迭代到上限步数. 实现的第二种方法是给定一个接受比率指标 R、迭

代步长上限 U 和下限 L，每一温度至少迭代 L 步且记录同一温度迭代的总次数和被接受的次数，当迭代超过 L 步时，若接受次数同总次数的比率不小于 R 时，在这一温度不再迭代而开始温度下降，否则，一直迭代到上限步数.

同样可以用拒绝次数为指标类似上面的讨论得到一些控制迭代步数的规则.

(3) 概率控制法

以概率 1 跳出任何一个局部最优解是概率控制法的一个基本思想. 设状态 i 是一个局部最优状态，$p_{ii}(t)$ 为 i 状态一步转移不动的概率，则 n 步保持不动的概率是 $p_{ii}^n(t)$. 于是温度 t 时状态 i 不转移的平均次数

$$\tilde{N}_i(t)=\sum_{n=1}^{\infty} n p_{ii}^n=\frac{p_{ii}}{(1-p_{ii})^2}\leqslant\frac{1}{(1-p_{ii})^2}.$$

当产生概率为(3.6)和接受概率为(3.7)时，

$$p_{ii}(t)=1-\frac{1}{|N(i)|}\sum_{\substack{j\in N(i)\\ j\neq i}}\min\left\{1,\exp\left[-\frac{f(j)-f(i)}{t}\right]\right\},$$

于是

$$\tilde{N}_i(t)\leqslant\frac{1}{(1-p_{ii})^2}=\frac{|N(i)|^2}{\left(\sum\limits_{\substack{j\in N(i)\\ j\neq i}}\min\left\{1,\exp\left[-\frac{f(j)-f(i)}{t}\right]\right\}\right)^2},$$

近似估计

$$\tilde{N}_i(t)\approx\frac{|N(i)|^2}{\left(\sum\limits_{\substack{j\in N(i)\\ j\neq i}}\min\left\{1,\exp\left[-\frac{f_{\max}-f_{\min}}{t}\right]\right\}\right)^2},\quad(3.51)$$

其中，$f_{\max}$，$f_{\min}$ 分别为整个状态空间的最大和最小费用值. 从(3.51)的估计可以看出：当温度较高时，跳出局部最优的平均次数为 $O(|N(i)|^2)$，这与(1)的估计基本吻合；当温度变低时，跳出局部最优的平均次数增加非常快，增加的速度与 $e^{\frac{1}{t}}$ 同阶. 由此可

以看出，在实际应用中，(3.53) 也是不适用的. 实际应用中，(1) 和 (2) 比较容易实现.

4. 算法的终止原则

模拟退火算法从初始温度开始，通过在每一温度的迭代和温度的下降，最后达到终止原则而停止. 尽管有些原则有一定理论的指导，终止原则大多是直观的. 下面分类讨论.

(1) 零度法

模拟退火的最终温度为零. 因而最为简单的原则是：给定一个比较小的正数 ε，当温度 $t_k \leqslant \varepsilon$ 时，算法停止. 表示已经达到最低温度.

(2) 循环总数控制法

这一简单原则在时齐算法的温度下降方法(2) 中已经提及，即总的温度下降次数为一定值 K，当温度迭代次数达到 K 时，停止运算. 这一原则可分为两类，一类是整个算法的总迭代步数为一个固定数，它表示各温度时马氏链迭代数(内循环) 的总和为一个给定的数. 这样很容易计算算法的复杂性，但需要合理分配内循环的长度和温度下降的次数. 另一类是内循环的次数由迭代长度规则的(2) 和(3) 决定，温度下降次数(外循环) 为一个定值. 这样的控制法对估计算法的复杂性有一定的困难. 解决的方法一是通过理论的估计，二则是类似迭代长度规则的(2) 给出每一温度的迭代长度上限.

(3) 基于不改进规则的控制法

在一个温度和给定的迭代次数内没有改进当前的局部最优解，则停止运算. 模拟退火的一个基本思想是跳出局部最优解. 直观的结论是在较高的温度没能跳出局部最优解，则在低的温度跳出最优解的可能也比较小. 由此产生上面的停止原则.

(4) 接受概率控制法

该方法与(3) 相同的思想. 给定一个指标 $\chi_f > 0$ 是一个比较

小的数，除当前局部最优解以外，其他状态的接受概率都小于 χ_f 时，停止运算. 实现(3) 和(4) 时，记录当前局部最优解，给定一个固定的迭代次数，当在规定的次数里没有离开局部最优解或每一次计算的接受概率都小于 χ_f，则在这个温度停止计算.

(5) 邻域法

若采用定理 3.13 的发生概率和(3.7) 的接受概率，且设 f_0 和 f_1 分别为一个邻域内的局部最优和次最优值，当满足

$$\exp\left(-\frac{f_1 - f_0}{t}\right) < \frac{1}{N} \tag{3.52}$$

时，其中 N 为邻域的大小，从局部最优到次优的接受概率满足(3.52)，而从局部最优到其他费用更高的状态的接受概率更小. 直观的想法是邻域中每次至少有一个状态被接受，但当满足(3.52)时，除局部最优解以外状态的接受概率都小于邻域总点平均数，此时可以认为从局部最优解转移到其他状态的可能性很小，因此停止. 通过(3.52) 可得终止温度

$$t_f \leqslant \frac{f_1 - f_0}{\log N}. \tag{3.53}$$

(6) Lundy 和 Mees 方法

Lundy 和 Mees[11] 从概率的意义给出一个判定方法. 给定充分小的正数 δ 和 ε，在达到终止温度应该满足

$$\Pr(X(k) = i \wedge f(i) > f_{\mathrm{OPT}} + \varepsilon \mid t = t_f) < \delta,$$

其中 $\wedge$ 表示逻辑的“与”. 由平稳分布所具有的性质，在温度 t，$\Pr(X(k) = i) = v_i(t)$，可以近似为

$$\Pr(X(k) = i \wedge f(i) > f_{\mathrm{OPT}} + \varepsilon \mid t = t_f) \approx \sum_{i: f(i) > f_{\mathrm{OPT}} + \varepsilon} v_i(t)$$

$$< (|D| - 1)\exp\left(-\frac{\varepsilon}{t}\right) < \delta,$$

直接推导可得终止温度

$$t_f \leqslant \frac{\varepsilon}{\log(|D| - 1) - \log\delta}. \tag{3.54}$$

(7) Aarts 和 Van Laarhoven 法

Aarts 和 Van Laarhoven[3] 在(3.6)和(3.7)的条件下，用概率分析的方法给出终止温度的精细估计，他们的基本思想是根据(3.40)期望费用

$$\langle f(t)\rangle = \sum_{i\in D} f(i)v_i(t)$$

的函数特性(定理 3.18)

$$\frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t} = \frac{1}{t^2}\sum_{i\in D}\{f(i) - \langle f(t)\rangle\}^2 v_i(t).$$

在终止温度 t_f，由泰勒(Taylor)展开的第一项，费用均值近似为

$$\langle f(t_f)\rangle - f_{\mathrm{OPT}} \approx t_f * \frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t}\bigg|_{t=t_f}. \tag{3.55}$$

当(3.55)的右端项充分小的时候，则左端的差值也较小，可以认为已达到要求. 用数学公式描述为：对给定的 $\varepsilon > 0$，当

$$\frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t}\bigg|_{t=t_f} \frac{t_f}{\langle f(t_0)\rangle} < \varepsilon \tag{3.56}$$

或

$$\frac{\mathrm{d}\langle f(t)\rangle}{\mathrm{d}t}\bigg|_{t=t_f} \frac{t_f}{\langle f(t_0)\rangle - \langle f(t_f)\rangle} < \varepsilon \tag{3.56'}$$

时，计算停止. (3.56)和(3.56′)可以用样本均值和样本矩近似替代

$$\frac{\langle f^2(t_f)\rangle - \langle f(t_f)\rangle^2}{t_f\langle f(t_0)\rangle} < \varepsilon, \tag{3.57}$$

$$\frac{\langle f^2(t_f)\rangle - \langle f(t_f)\rangle^2}{t_f(\langle f(t_0)\rangle - \langle f(t_f)\rangle)} < \varepsilon, \tag{3.57'}$$

当满足(3.57)或(3.57′)时，计算停止. 用样本值近似

$$\frac{\overline{f^2(t_f)} - \left(\overline{f(t_f)}\right)^2}{t_f\,\overline{f(t_0)}} < \varepsilon, \tag{3.58}$$

$$\frac{\overline{f^2(t_f)} - \left(\overline{f(t_f)}\right)^2}{t_f\left(\overline{f(t_0)} - \overline{f(t_f)}\right)} < \varepsilon. \tag{3.58'}$$

理论上是用一个马尔可夫链描述模拟退火算法的变化过程，因此具有全局最优性. 实际应用中的模拟退火算法是一个启发式算法. 它有诸多的参数需要调整，如起始温度、温度下降的方案、固定温度时的迭代长度及终止规则等，这样需要人为地调整. 人为的因素，如对问题的了解、参数和规则的搭配等，造成计算结果的差异. 解决这个矛盾的方法主要通过大量的数值模拟计算，从中选择比较好的参数搭配.

3.6　应用案例 —— 下料问题

下料问题存在于诸多如玻璃、钢板、木材、纸张和制衣等的裁剪中. 它们的共同特点是原料的尺寸大于需求的尺寸，而需求的品种尺寸可以不相同，最终的目标是在满足需求的前提下，使得边角废料最小. 在以上所提出的问题中，它们都是二维参数. 原料有其长度和宽度，需求的品种同样有长和宽的要求. 本节讨论的问题产生于新闻界的锌版下料问题. 一个印刷厂负责若干家报刊的印刷工作，它要将锌版按报刊版面的不同要求下料. 锌版原料的长和宽基本相同，我们假设它们都相同. 报刊版面则有不同的长宽.

我们首先介绍常见的几种直线切割(guillotine cutting)的下料方法. 在不混淆的前提下，我们称锌版原料为原料.

这种切割保持切割线平行于原料的一个边线，对截开的料继续采用这个规则，一直到满足需求为止. 参考图 3.2 的三类直线切割法.

三种分类的主要区别：2 级切割先将原料按一个方向切割后，再对切割后的条材与原切割方向垂直切割；3 级切割是在 2 级切割后的料上再垂直于第二次切割方向切割；多于 3 级切割的直线切

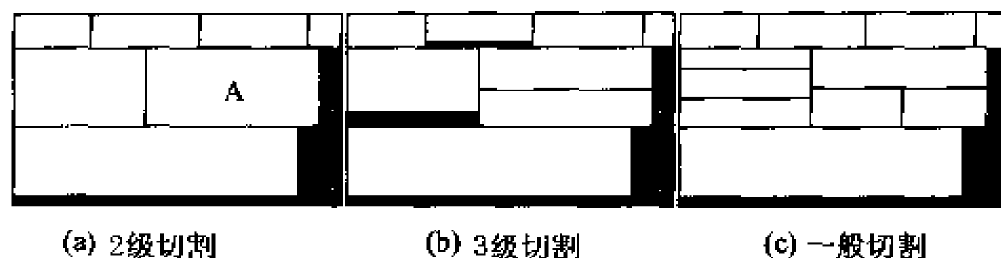


图 3.2 直线切割分类

割为一般切割.2 级切割、3 级切割和一般切割的区别见图 3.2(a)(b)(c) 中 A 区域所示.图中黑色部分表示剩余废边角料.

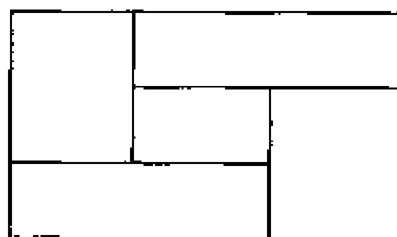


图 3.3 套裁方式

更一般的套裁方式如图 3.3.

直观可知,套裁节约用料.平面上的套裁问题是二维装箱问题,是一维装箱问题的推广,因此,这个优化问题是 NP-hard.文献[13]中采用模拟退火算法求解下料套裁问题,文中把下料的位置用两个坐标点表示,如原料的左下角和右上角的坐标分别为$(x0,y0)$,$(x0+x1,y0+y1)$,于是第一个矩形为$R_1=[(x0,y0),(x0+x1,y0+y1)]$,所有的下一个下料点都是以剩余矩形的左下角开始.

若所下料 $a1$ 的长宽尺寸分别为 $x2$ 和 $y2$,则下料后的剩余矩形为,如图 3.4,

$$\begin{aligned}&[(x0,y0),(x0+x1,y0+y1)]\\&-[(x0,y0),(x0+x2,y0+y2)]\\&=\{[(x0+x2,y0),(x0+x1,y0+y1)]\\&\quad[(x0,y0+y2),(x0+x1,y0+y1)]\},\end{aligned}\tag{3.59}$$

按顺时针记录矩形,则有

$$R2=[(x2+x0,y0),(x0+x1,y0+y1)],$$
$$R3=[(x0,y2+y0),(x0+x1,y0+y1)].$$

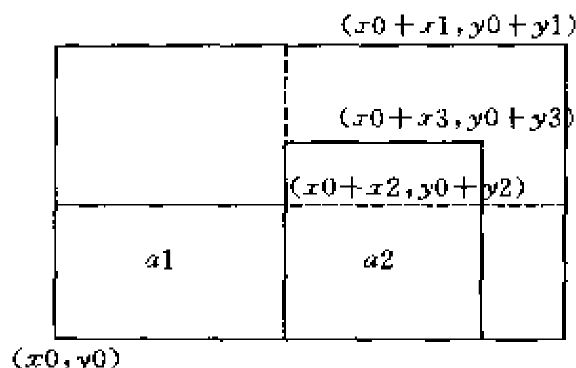


图 3.4　下料运算图

若需求产品的长宽尺寸分别为 $x2$ 和 $y2$,将料下在右上角则

$$\begin{aligned}&[(x0,y0),(x0+x1,y0+y1)]\\&-[(x0+x1-x2,y0+y1-y2),(x0+x1,y0+y1)]\\&=\{[(x0,y0),(x0+x1-x2,y0+y1)],[(x0,y0),\\&\quad(x0+x1,y0+y1-y2)]\}.\end{aligned}\tag{3.60}$$

(3.59)和(3.60)分别给出按左下角和右上角下料的余料计算公式.在公式中,希望考虑尽量大的矩形块,因此它们有重复部分.任意两个矩形 $[(x0+x1,y0+y1),(x0+x2,y0+y2)]$ 和 $[(x0+x3,y0+y3),(x0+x4,y0+y4)]$ 重复部分的计算公式为

$$\begin{aligned}&\mathrm{Com}([(x1,y1),(x2,y2)],[(x3,y3),(x4,y4)])\\&=[(\max\{x1,x3\},\max\{y1,y3\}),(\min\{x2,x4\},\min\{y2,y4\})].\end{aligned}\tag{3.61}$$

在余下的矩形块中,再安排下一次的下料.当下料的次数变多,可选择的矩形块渐多.

第一次下料后,出现两块矩形.若再进行一次下料,必须从中选定一块.当选中一块后,采用一个启发式的方法重新分块.基本思想是将未选用的块用(3.60)减去公共部分,如图 3.4 中的 $[(x0,y0+y2),(x0+x1,y0+y1)]$ 块未选中,则重新分块中的一块为

$[(x0,y0+y2),(x0+x1,y0+y1)]-[(x0+x2,y0+y2),$
$(x0+x1,y0+y1)]$
$=\{[(x0,y0+y2),(x0+x2,y0+y1)],[(x0,y0+y2),$
$(x0+x1,y0+y2)]\}$
$=[(x0,y0+y2),(x0+x2,y0+y1)].$

选用的块采用(3.59)的运算关系下料. 如图 3.4 中$[(x0+x2,y0),(x0+x1,y0+y1)]$块被选中下料,所下料的长宽分别为 $x3$ 和 $y3$,记为图 3.4 的 $a2$,则在 $R2$ 中用(3.59)的运算重复计算.

$$
\begin{aligned}
&[(x0+x2,y0),(x0+x1,y0+y1)] \\
&\quad -[(x0+x2,y0),(x0+x2+x3,y0+y3)] \\
&=\{[(x0+x2+x3,y0),(x0+x1,y0+y1)], \\
&\quad [(x0+x2,y0+y3),(x0+x1,y0+y1)]\}.
\end{aligned}
$$

记

$$
\begin{aligned}
R4&=[(x0+x2+x3,y0),(x0+x1,y0+y1)], \\
R5&=[(x0+x2,y0+y3),(x0+x1,y0+y1)].
\end{aligned}
$$

在第 K 次切割完成后,一共剩余 $K+1$ 个矩形,其中,$K-1$ 个除边界外互不相交,有两块有公共相交部分. 第 $K+1$ 次切割从 $K+1$ 个矩形中选出哪一个稍后讨论. 选出的矩形有两种可能性. 第一是从不相交的 $K-1$ 个矩形中选出,此时,相交的那两块将重合部分归面积大的那个矩形,因此又分为两个不重合的矩形. 第二是从相交的一对矩形中选择一个,原有重合部分归属选择上的那个矩形,因此原有重合部分的料又分为两个不重合的矩形. 在任何一个矩形下一次料,由图 3.4 知,必然余下两块有重合部分的矩形.

采用上述的方法,每次下料前,都有若干个独立的矩形和一对带有重复的相交矩形. 以上的规则可以重复运用. 为了以后的方便,我们将需求料称为产品,余下的每一个矩形称为矩形原料.

任何一个原料和产品都可以用以上形式的两个顶点坐标表示. 按(3.59),(3.60)和(3.61)的运算,K 次料后,有 $K-1$ 个不相交的矩形(边界可以相连)和两个有相同公共部分的矩形,如此共计 $K+1$ 个矩形. 于是,如何从 $K+1$ 个矩形中选下一个下料成为要解决的问题.

如果将产品的下料顺序看成问题的一个解,现在必须给出如何从诸多个矩形中选择一个,及从何处下料. 当这些问题解决后,就可以计算余料的大小,即目标值. 优化问题就是确定一个最优的下料序. 用模拟退火求解时,其邻域的结构可以模仿 TSP 中的 2-opt 来构造,交换两个产品的加工顺序. 于是问题解的形式和邻域结构已经得到.

如何选择矩形原料下料的思想类似于第 2.3.2 节例 2.11 的目标函数替代法中解的构造. 在给定的一个下料序后,不求解这个下料序的最优下料方法(若求解,相当于对原问题求最优下料方法),用一种启发式方法安排下料,在产品完全排完后,就可以计算余料的值,计为目标值.

选择矩形原料的启发式规则是根据等待下料的产品,采用下列方案.

(1) 首选,产品的长或宽同原料的长或宽不超过一个给定的比较小的正数 ε. 有这样的可能,原料宽同产品长的差不超过一个给定的比较小的正数 ε,如图 3.5(a);其二是原料长同产品长的差不超过一个给定的比较小的正数 ε,如图 3.5(b);图 3.5 中的灰色部分表示产品的下料位置和大小.

(2) 其次比较原料的长宽之和同产品长宽之和的差值的大小,从中选差值最小的原料.

在多次下料后,有些原料因为尺寸太小而失去使用价值,此时再记录这样小的矩形原料只会增加内存而无任何好处. 因此,应该删除这样的矩形原料. 对给定的数 $\varepsilon>0$,在已下 K 块产品后,若余

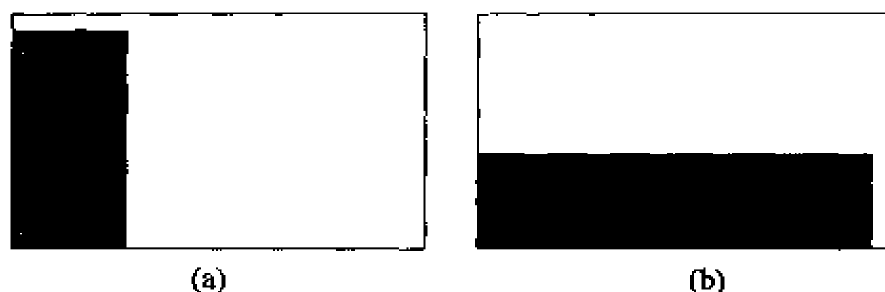


图 3.5 产品与原料吻合

下的 $K+1$ 块矩形原料中有长或宽不超过 ϵ 的，则删除这些原料块.

目标函数采用余料和同总用料的比率. 由此可以估计目标值的上界是 100%，即所有的料被浪费，下界是 0%，表示无浪费. 于是，可以估计上下界差(3.37)不超过 $\delta=1$. 起始温度用(3.37)计算，$t_0=K\delta$，其中 $K=100$.

温度的下降采用 Lundy 和 Mees 的(3.49)

$$t_{k+1}=t_k(1+\beta t_k)^{-1},$$

其中，

$$\beta=\frac{t_0-t_f}{Mt_0t_f},$$

t_f 为一给定的值，M 是温度可下降的最大次数. 在文献[13]中，作者采用非时齐算法，即每一温度只迭代一步. 因此，总的迭代次数为 M.

计算的数据采用随机产生和香港地区一个印刷厂的实际数据两类. 模拟退火的参数分别为：起始温度 $t_0=100$，$t_f=10$，$M=10000$. 除了最多迭代 M 步的停止准则以外，再加上如果边角余料之和不超过开始时原料的 5%，则停止计算. 在此，我们仅采用随机产生数据的计算结果.

数据产生的参数是：原料分两种尺寸：$400\times200400\times400$，产品料的种类有 $5\sim35$ 个种类，每一类产品料对两种尺寸各生成

50个随机数据组. 产生的原则是每组数据一定有余料为零的最优解. 整个计算是在主频33Hz的486上实现. 结果见表3.2和表3.3.

表3.2 原料400×200的模拟结果

产品料种类	测试数据组数	平均计算时间(s)	达到最优解的组数	平均的迭代步数
5	50	3.8	50	8
10	50	13.0	50	15
15	50	42.0	50	303
20	50	101.5	50	868
25	50	405.2	46	857
30	50	413.0	44	866
35	50	445.3	41	943

表3.3 原料400×400的模拟结果

产品料种类	测试数据组数	平均计算时间(s)	达到最优解的组数	平均的迭代步数
5	50	2.3	50	4
10	50	9.3	50	12
15	50	44.0	50	230
20	50	221.0	47	522
25	50	389.3	45	855
30	50	432.1	41	941
35	50	442.0	42	921

从表 3.2 和表 3.3 可以看出，达到最优解的组数相对较高，算法停止时的平均迭代步数没有超过所给的上限 M，但计算时间随产品料种类数增加而增加的速度较快. 因此，可以说算法的计算效果还是相当好的.

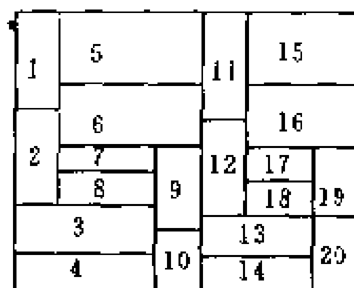


图 3.6　400 × 400 料 20 个产品的最优下料图

图 3.6 表示一个 20 个产品种类在 400 × 400 的原料的最优下料图，其余料率为 0%.

图 3.7 描述对图 3.6 原料和产品料用模拟退火计算时，迭代步数同余料率之间的关系，其中迭代步数的每一格表示 50 个迭代步数.

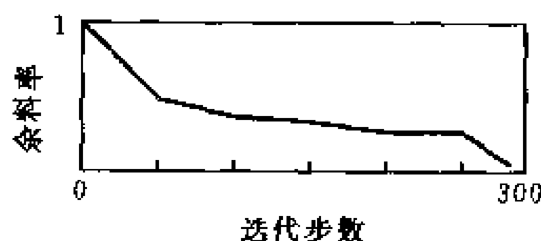


图 3.7　余料率同迭代步数关系

以平均的意义从图 3.7 可以看出，余料率同迭代步数的关系是：随迭代的步数增加，余料率逐步下降，在 300 时基本达到零余料率.

练　习　题

1. 对非能量最低的状态，证明：存在 $t > 0$，使得其波兹曼概率在 $(0,t)$ 单调升.

2. 若 $v_j = \lim\limits_{n\to\infty} p_{ij}^{(n)} = \lim\limits_{n\to\infty} \Pr(X(n) = j \mid X(0) = i)\ (j \in \Omega)$ 的

极限有限，且状态空间有限，证明：

$$v_j = \sum_{i=1}^{|\Omega|} v_i p_{ij}^{(n)} = \sum_{i=1}^{|\Omega|} v_i p_{ij}.$$

3. 时齐马氏链的一步转移矩阵为 $\boldsymbol{P} = (p_{ij})$，$p_{ij} = \Pr(X(k+1) = j \mid X(k) = i)$. 它的 n 部转移概率为

$$p_{ij}^{(n)} = \Pr(X(n) = j \mid X(0) = i).$$

它的矩阵形式为 $\boldsymbol{P}^{(n)} = (p_{ij}^{(n)})$. 证明：$\boldsymbol{P}^{(n)} = \boldsymbol{P}^n$.

4. 若发生概率

$$G_{ij}(t) = \begin{cases} \dfrac{1}{L}, & \forall\, j \in N(i), \\ 0, & j \notin N(i), \end{cases}$$

其中 L 为一个正数. 接受概率按(3.7)定义. 证明：当邻域的定义使得状态空间连通时，由此定义的时齐马氏链有平稳分布，写出平稳分布的表达式.

5. 若接受概率满足定理 3.10 的条件，发生概率按文献[3]中的定义

$$\exists\ |D| \times |D| \text{ 矩阵 } \boldsymbol{Q} \text{ 满足}$$

$$\forall\, i, j \in D, \text{使得 } Q_{ij} = Q_{ji};$$

$$\forall\, i \in D, j \in N(i), \text{使得 } G_{ij} = \frac{Q_{ij}}{\sum_{l \in D} Q_{il}},$$

证明：它的平稳分布存在且为

$$v_i(t) = \frac{\left(\sum_l Q_{il}\right) A_{i_0 i}(t)}{\sum_{j \in D} \left(\sum_l Q_{jl}\right) A_{i_0 j}(t)}, \backslash\ \forall\, i \in D.$$

6. 若发生概率按练习 4 定义，文献[4]中给出接受概率为

$$A_{ij}(t) = \left(1 + \exp\left(-\frac{f(j) - f(i)}{t}\right)\right)^{-1}.$$

证明：它的平稳分布存在且为

$$v_i(t)=\frac{\exp\left(-\dfrac{f(i)-f_{\mathrm{OPT}}}{t}\right)}{\sum\limits_{j\in D}\exp\left(-\dfrac{f(j)-f_{\mathrm{OPT}}}{t}\right)},\quad \forall\, i\in D.$$

7. 验证

$$v_i(t_k)=\frac{\exp\left(-\dfrac{f(i)-f_{\mathrm{OPT}}}{t_k}\right)}{\sum\limits_{j\in D}\exp\left(-\dfrac{f(j)-f_{\mathrm{OPT}}}{t_k}\right)},\quad \forall\, i\in D.$$

满足(3.27).

8. 若 $A(t)$ 和 $G(t)$ 满足定理 3.13 的条件,且 $A(t)$ 和 $G(t)$ 分别满足(3.6)和(3.7),则当 $i=i_{\mathrm{OPT}}$ 时,平稳分布 $v_i(t)$ 为 t 的单调减函数.

9. 在时齐马氏链平稳分布存在的条件下,若 $A(t)$ 是(3.7)的形式,证明

$$\frac{\mathrm{d}}{\mathrm{d}\log t}\langle f(t)\rangle=\frac{\langle f^2(t)\rangle-\langle f(t)\rangle^2}{t}.$$

10. 你对 3.6 节下料问题的模拟退火算法有什么样的评价?按照你的观点实现模拟退火算法,并同 3.6 节的算法比较.

11. 比较禁忌搜索、模拟退火算法在下料问题的应用效果.

12. 用模拟退火求解 TSP,同禁忌搜索算法比较它们的计算效率.

13. 对你感兴趣的优化问题,尝试模拟退火的计算效果.

参 考 文 献

1. Metropolis N, Rosenbluth A, Rosenbluth M et al. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 1953, 21:1087～1092

2. Kirkpatrick S, Gelatt Jr C D, Vecchi M P. Optimization by simulated

annealing. Science, 1983, 220:671～680

3. Aarts E H L, van Laarhoven P J M. Simulated Annealing: Theory and Application. Dordrecht: D Reidel Publishing Company, 1987
4. Ackley D H, Hinton G E, Seinowski T J. A learning algorithm for Boltzmann machines. Cognitive Science, 1985, 9:147～169
5. Seneta E. Non-negative Matrices and Markov Chains. 2nd Edition. New York: Springer Verlag, 1981
6. Isaacson D, Madsen R. Makov Chains. New York: Wiley, 1976
7. Geman S, Geman D. 1984, Stochastic relaxation, Gibbs distributions, and the Bayaesian restoration of images. In: IEEE Proc. Pattern Analysis and Machine Intelligence, PAMI-6, 1984. 721～741
8. Anily S, Federgruen. Probabilistic Analysis of Simulated Annealing Methods. Graduate School of Business, Columbia University, New York, Preprint, 1985
9. Hajek B. Cooling Schedules for Optimal Annealing, Working Paper. (MOR86), 1986
10. 盛骤. 概率论与数理统计. 第二版. 北京: 高等教育出版社, 1990
11. Lundy M, Mees A. Convergence of an annealing algorithm. Mathematical Programming, 1986, 34:111～124
12. Huang M D, Romeo F, Sangiovanni-Vincentelli A L. An efficient general cooling schedule for simulated annealing. In: Proceeding of IEEE Int. Conference on Computer-Aided Design. Santa Clara. 1986. 381～384
13. Lai K K, Chan J W M. Developing a simulated annealing algorithm for the cutting stock problem, Computers Ind. Engng, 1996

第4章

遗 传 算 法

遗传算法(genetic algorithms)是在70年代初期由美国密执根(Michigan)大学的Holland教授发展起来的. 1975年,Holland发表了第一本比较系统论述遗传算法的专著《Adaptation in Natural and Artificial Systems[1]》. 本章先介绍遗传算法及一些基本性质,然后讨论遗传算法实现的技术问题,最后通过在约束批量模型的应用了解算法实现过程.

4.1 遗 传 算 法

遗传算法主要借用生物进化中"适者生存"的规律."适者生存"揭示了大自然生物进化过程中的一个规律:最适合自然环境的群体往往产生了更大的后代群体. 因此有必要在介绍遗传算法之前,先简单了解生物进化的基本过程,见图4.1.

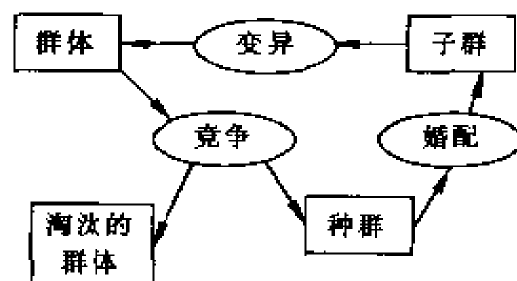


图 4.1 生物进化循环图

以这个循环圈的群体(population)为起点,经过竞争后,一部

分群体被淘汰而无法再进入这个循环圈，而另一部分则成为种群(reproduction)．优胜劣汰在这个过程中起着非常重要的作用，这在自然界显得更加突出．因为自然天气的恶劣和天敌的侵害，大自然中的很多动物的成活率是非常低的．即使在成活群体中，还要通过竞争产生种群．种群通过婚配的作用产生子代群体(简称子群)．进化的过程中，可能会因为变异而产生新的个体．综合变异(mutation)的作用，子群成长为新的群体而取代旧群体．在新的一个循环过程中，新的群体将替代旧的群体而成为循环的开始．

以人类进化为例，人类群体在自然竞争、淘汰后，形成了繁殖后代的群体，即成人群体．群体的每一个体是一个人，每个人包含有 46 条染色体(chromosome)，组成 23 对同源染色体(参见[2]，P.8)．男女性的结合使得对应的 23 对染色体优胜劣汰再产生 23 对染色体，因而形成一个新的生命．以性别染色体为例(见图 4.2)，母亲的性染色体中由 X(1) 和 X(2) 基因(gene) 组成，父亲的性染色体由X(*m*)和Y基因组成．两个染色体的结合是这两对基因竞争后的产物，竞争的结局可能是{X(1),X(*m*)}(女)，{X(1),Y}(男)，{X(2),X(*m*)}(女)，{X(2),Y}(男)，这一对新的结合物为子代的一对性染色体．每一条染色体是由特定的基因组成．在自然进化中，会出现某些基因的变异，形成一个新的个体．

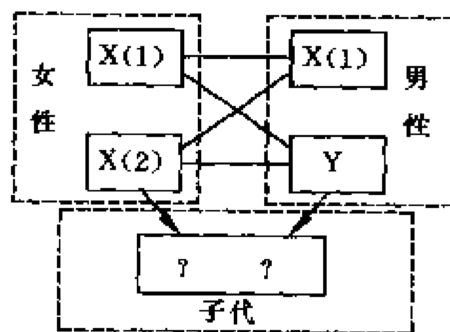


图 4.2　性别染色体的竞争

遗传算法主要借鉴了生物进化的一些特征，其主要特征体现为：

- 进化发生在解的编码上．这些编码按生物学的术语称为染色体．由于对解进行了编码，优化问题的一切性质都通过

编码来研究.编码和解码是遗传算法的一个主题.

- 自然选择规律决定哪些染色体产生超过平均数的后代.遗传算法中,通过优化问题的目标而人为地构造适应函数以达到好的染色体产生超过平均数的后代.
- 当染色体结合时,双亲的遗传基因的结合使得子女保持父母的特征.
- 当染色体结合后,随机的变异会造成子代同父代的不同.

遗传算法包含以下的主要处理步骤.首先是对优化问题的解进行编码,此处,我们称一个解的编码为一个染色体,组成编码的元素称为基因.编码的目的主要是用于优化问题解的表现形式和利于之后遗传算法中的计算.第二是适应函数的构造和应用.适应函数基本上依据优化问题的目标函数而定.当适应函数确定以后,自然选择规律是以适应函数值的大小决定的概率分布来确定哪些染色体适应生存,哪些被淘汰.生存下来的染色体组成种群,形成一个可以繁衍下一代的群体.第三是染色体的结合.双亲的遗传基因结合是通过编码之间的交配(crossover)达到下一代的产生.新一代的产生是一个生殖过程,它产生了一个新解.最后是变异.新解产生过程中可能发生基因变异,变异使某些解的编码发生变化,使解有更大的遍历性.表 4.1 列出了生物遗传基本概念在遗传算法中作用的对应关系.

表 4.1 生物遗传概念在遗传算法中的对应关系

生物遗传概念	遗传算法中的作用
适者生存	在算法停止时,最优目标值的解有最大的可能被留住
个体(individual)	解
染色体(chromosome)	解的编码(字符串,向量等)
基因(gene)	解中每一分量的特征(如各分量的值)

续表

生物遗传概念	遗传算法中的作用
适应性(fitness)	适应函数值
群体(population)	选定的一组解(其中解的个数为群体的规模)
种群(reproduction)	根据适应函数值选取的一组解
交配(crossover)	通过交配原则产生一组新解的过程
变异(mutation)	编码的某一个分量发生变化的过程

最优化问题的求解过程是从众多的解中选出最优的解. 生物进化的适者生存规律使得最具有生存能力的染色体以最大的可能生存. 这样的共同点使得遗传算法可以在优化问题中应用. 我们以一个简单的例子来理解表 4.1.

例 4.1 用遗传算法求解 $f(x)=x^2, 0\leqslant x\leqslant 31$, x 为整数的最大值.

一个简单的表示解的编码是二进制编码, 即 0,1 字符串. 由于变量的最大值是 31, 因此可以采用 5 位数的二进制码. 如

$$10000\rightarrow 16 \quad 11111\rightarrow 31 \quad 01001\rightarrow 9 \quad 00010\rightarrow 2,$$

以上的 5 位字符串称为染色体. 每一个分量称为基因, 每个基因有两种状态 0 或 1. 模拟生物进化, 首先要产生一个群体, 可以随机取 4 个染色体组成一个群体, 如 $x_1=(00000)$, $x_2=(11001)$, $x_3=(01111)$, $x_4=(01000)$. 群体有 4 个个体. 适应函数可以依据目标函数而定, 如适应函数 $\text{fitness}(x)=f(x)=x^2$. 于是

$$\text{fitness}(x_1)=0,\ \text{fitness}(x_2)=25^2,$$
$$\text{fitness}(x_3)=15^2,\ \text{fitness}(x_4)=8^2.$$

定义第 i 个个体入选种群的概率为

$$p(x_i)=\frac{\text{fitness}(x_i)}{\sum_j \text{fitness}(x_j)}.$$

于是, 适应函数值大的染色体个体的生存概率自然较大. 若群体中

选 4 个个体成为种群，则极有可能竞争上的是 $x_2=(11001)$，$x_2=(11001)$，$x_3=(01111)$，$x_4=(01000)$. 若它们结合，采用如下的交配方式，称为简单交配

$$\left.\begin{aligned} x_2 &= (11|001) \\ x_3 &= (01|111) \end{aligned}\right\} \longrightarrow \begin{aligned} y_1 &= (11|111) \\ y_2 &= (01|001) \end{aligned}$$

$$\left.\begin{aligned} x_2 &= (11|001) \\ x_4 &= (01|000) \end{aligned}\right\} \longrightarrow \begin{aligned} y_3 &= (11|000) \\ y_4 &= (01|001) \end{aligned}$$

即交换第二个位置以后的基因，得到 y_1, y_2, y_3 和 y_4. 若 y_4 的第一个基因发生变异，则变成 $y_4=(11001)$. □

通过例 4.1，我们可以将求解组合优化问题的遗传算法简化地描述为：

遗传算法

STEP1　选择问题的一个编码；给出一个有 N 个染色体的初始群体 pop(1)，$t:=1$；

STEP2　对群体 pop(t) 中的每一个染色体 $\text{pop}_i(t)$ 计算它的适应函数

$$f_i = \text{fitness}(\text{pop}_i(t));$$

STEP3　若停止规则满足，则算法停止；否则，计算概率

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \quad i=1,2,\cdots,N, \tag{4.1}$$

并以概率分布(4.1)从 pop(t) 中随机选一些染色体构成一个种群

$$\text{newpop}(t+1) = \{\text{pop}_j(t) \mid j=1,2,\cdots,N\};$$

[注]newpop($t+1$) 集中可能重复选 pop(t) 中的一个元素，如例 4.1 中的 x_2 就选取两次.

STEP4　通过交配，交配概率为 P_c，得到一个有 N 个染色体的 crosspop($t+1$)；

STEP5　以一个较小的概率 p，使得一个染色体的一个基因发生变异，形成 mutpop$(t+1)$；$t := t+1$，一个新的群体 pop(t) = mutpop(t)；返回 STEP2.

种群的选取方式(4.1)称为轮盘赌. 以下再通过例 4.2 理解遗传算法.

例 4.2　用遗传算法求 $\max f(x) = 1 - x^2,\ x \in [0,1]$.

由于对连续变量求解，要解决的一个问题是如何编码. 假设对解的误差要求是 1/16，则可以采用 4 位二进制编码. 对应关系为

$$(abcd) \leftrightarrow \frac{a}{2} + \frac{b}{4} + \frac{c}{8} + \frac{d}{16}.$$

若计算的一步如表 4.2.

表 4.2　遗传算法中的一步计算

x 值	旧群体 pop(1)	适应函数值 f	概率分布 p	new pop	交配位	cross pop	是否变异	mut pop	x 值
1/16	0001	0.996	0.318	0001	00|01	0000	N	0000	0
1/4	0100	0.938	0.299	0100	01|00	0101	Y	1101	13/16
3/16	0011	0.965	0.308	0001	000|1	0001	N	0001	1/16
7/8	1110	0.234	0.075	0011	001|1	0011	N	0011	3/16
平均值 = 0.7833，交配概率 $p=1$，变异概率 $p_m = 0.02$									

在表 4.2 中，newpop 只选择 4 个染色体，交配方法采用随机选择一对染色个体按交配位进行交叉，如 0001 和 0100 的交配位随机选在位置 2，经过交配产生 0000 和 0101 两个个体. 在算法中，0001 和 0011 的交配位随机选在位置 3，经过交配产生 0001 和 0011 两个个体. 这样的交配称为简单交配. 交配后的染色体组成 crosspop. 在 crosspop 中可能有些染色体会产生变异. 简单的变异

是 crosspop 中每一个基因都以同样一个概率变异. 如 0101 第一位的基因发生变异. 变异后，得到 mutpop，也就是新的群体. □

上面的例子是简单遗传算法计算的一步. 简单遗传算法可以理解为：求解的问题是极大目标函数的优化问题；采用 0-1 二进制编码；pop(t) 中的染色体个数是一个常数；初始群体随机选取；适应函数为目标函数；按轮盘赌方法选取染色体个数同 pop(t) 相同的种群；交配按例 4.2 的常规交配方法 —— 一对染色体按随机位交换位后的基因；染色体中的每一个基因都以相同的概率变异.

以上只对遗传算法有一个直观的了解，一些技术问题将在以后的各节中讨论. 归结起来有如下主要因素有待研究.

(1) 解的编码和解码. 遗传算法的基础工作之一是解的编码，只有在编码之后才可能有其他的计算. 例 4.1 和例 4.2 采用二进制编码，后续章节还会介绍其他一些基于问题的编码方式. 编码和解码是相对应的. 算法的最后一个工作是通过解码得到问题的一个解.

(2) 初始群体的选取和计算中群体的大小. 一般采用随机产生初始群体或通过其他方法先构造一个初始群体. 通过其他方法构造的初始群体可能会节省进化的代数，但也可能过早地陷入局部最优群体中. 我们称过早地陷入局部最优群体中的现象为早熟(premature) 现象. 群体中个体的个数称为群体的维数. 群体的维数越大，其代表性越广泛，最终进化到最优解的可能性越大. 但维数大的群体势必造成计算时间的增加，这又是我们不希望的. 群体的维数常常采用一个不变的常数，在一些应用中，群体的维数可以采用同遗传代数有关的变量，以使算法更有效.

(3) 适应函数的确定. 一般情况，适应函数同目标函数相关，以保证较优的解有较大的生存机会. 同第 2 章的例 2.7，可以采用替代函数，这将在后续部分讨论.

(4) 三个算子. 遗传算法的三个算子是：种群选取、交配和变

异或称突变. 新群体产生中的一个主要问题是如何选取种群. 上面的遗传算法中已经介绍,种群是以一个概率分布 —— 轮盘赌的形式选择个体而产生的. 种群选定后需考虑它的交配规则. 交配规则较多,这在后续部分将详细讨论,如双亲遗传法,主个体副个体规则,单亲遗传规则等. 还需考虑交配位的选取、交配概率及产生后代等一些细节问题. 变异是扩大染色体选择范围的一个手段. 通常,遗传算法实现变异的方法是赋予每一个基因一个相对比较小的变异概率. 通过随机模拟而决定一个基因是否变异. 变异概率过小使解有一定的局限性,遍历性较差. 变异概率较大使得进化的随机性增大,也不容易得到稳定的解. 因此,不可忽略变异概率如何确定这一过程.

遗传算法的优越性可以简单地归结为以下三条.

(1) 遗传算法适合数值求解那些带有多参数、多变量、多目标和在多区域但连通性较差的 NP-hard 优化问题. 对多参数、多变量的 NP-hard 优化问题,通过解析求解或是计算求最优解的可能性很小,主要依赖于数值求解. 遗传算法是一种数值求解的方法,是一个有普适性的方法,对目标函数的性质几乎没有要求,甚至都不一定要显式地写出目标函数,因此用遗传算法求解优化问题不足为奇. 遗传算法所具有的特点是记录一个群体,它可以记录多个解而不同于局部搜索、禁忌搜索和模拟退火仅仅是一个解,这多个解的进化过程正好适合于多目标优化问题的求解.

(2) 遗传算法在求解很多组合优化问题时,不需要有很强的技巧和对问题有非常深入的了解. 如排序(scheduling)、路线调度(routing) 问题、布局(layout) 问题等,这些问题可以参考第 2 章 2.4.2 的车间排序问题、第 1 章例 1.2 的 TSP 和第 3 章 3.6 的下料问题,如果不用这些普适性的算法(如禁忌搜索、模拟退火和遗传算法等)而采用其他的针对问题而设计的算法,要得到一个比较好的解,其算法的设计技巧非常强. 遗传算法在给问题的决策变量

编码后,其计算过程是比较简单的,且可以较快得到一个满意解.

(3) 遗传算法同求解问题的其他启发式算法有较好的兼容性. 如可以用其他的算法求初始解;在每一群体,可以用其他的方法求解下一代新群体.

遗传算法也不可避免的有它的不足. 如:

(1) 编码不规范及编码存在表示的不准确性. 如在例 4.2 中,是否有更好的编码方式表示连续变量就值得研究. 对 TSP 问题,一个解可以表示成城市的一个序列,如果就采用这种编码形式,例 4.2 的交配规则又无法采用.

例 4.3 假设 8 个城市的 TSP 问题, 两个染色体为 (14562387) 和(56327841),这两个交配后,得到

$$\begin{matrix}(145623|87) \\ (563278|41)\end{matrix} \longrightarrow \begin{matrix}(14562341) \\ (56327887)\end{matrix}$$

交配后的两个后代已不是 TSP 的一个可行解. □

于是,交配原则同问题的编码紧密相连.

(2) 单一的遗传算法编码不能全面地将优化问题的约束表示出来. 这个缺点类似于模拟退火算法 3.5.1 中有关解表示形式中的讨论. 考虑约束的一个方法就是对不可行解采用罚值,毫无疑问,计算的时间必然增加.

(3) 是否能保证收敛到最优解?这个问题需要讨论算法的一些理论,我们将在下一节讨论.

本章在没有特殊说明时所讨论的内容都是基于组合优化问题

$$\begin{aligned} z = \max f(\boldsymbol{x}) \\ \text{s.t. } g(\boldsymbol{x}) \geq 0, \\ \boldsymbol{x} \in D, \end{aligned}$$

其中 D 是有限离散定义域.

4.2　模板理论

模板理论(schema theorem)是针对4.1节的简单遗传算法建立的，主要从一种结构的角度说明遗传算法的收敛性. 这种结构，在此称为模板(schema). 在遗传算法的进化过程中，含有这种模板的个体具有较大的遗传概率[3]. 另一种类似模拟退火算法的基于马氏链的理论对遗传算法收敛性的研究将在4.3节介绍[4]. 简单遗传算法的主要特征有：群体和种群的维数相等，且不随代数的变化而变化；适应函数直接选用目标函数；种群中的个体通过轮盘赌(4.1)的方式选取；种群中的一对个体采用随机交配位的方式产生一对子代；每一个基因有相同的变异概率.

为了易于理解，我们将染色体称为向量. 一个给定的向量结构，包括关心的分量位置和值，称为该向量的一个模板. 如对下面的两个染色体，1010001 和 1111010，它们有共同的基因和结构 1 * 1 * 0 * * ，其中，标记 * 的位置表示基因不同或我们不关心这个位置的取值. 称 $H = 1 * 1 * 0 * *$ 为一个模板. 具有相同模板预示着两个染色体具有一些共同的性质. 对于 n 维的向量，如果采用二进制编码，共有 3^n 个模板. 从遗传学的角度，生物体的代代遗传，使得某些基因和模板得到生存，生存的一个原因是这些基因和模板有很强的适应能力.

若向量的各分量取值0,1或 * ，一个模板中0或1所占的位置称为模板位置，* 所占位置称为非模板位置. 模板的长度(length)定义为：从第一个模板位置到最后一个模板位置的所有分量个数减1. 如 $H = 1 * 1 * 0 * *$ 的长度为4，其含义是从模板位置1到最大的模板位置5有4个位置的距离. 模板长度记为 $\delta(H)$. 模板的阶数(order)定义为：模板位置对应的确定分量个数. 如 1 * 1 * 0 * * 的阶数为3. 记成 $o(H)$. 若染色体Y在 H 的模板位置

上对应的分量相同，则称 Y 具有 H 模板. 记 $G(t)$ 是第 t 代群体，即第 t 代染色体集合，

$$T(H,G(t)) = \{Y \in G(t) \mid Y \text{ 具有模板 } H\}, \tag{4.2}$$

其中 t 称为遗传的第 t 代. $T(H,G(t))$ 为模板 H 所包含的 $G(t)$ 中的所有染色体集合.

在下面的结论中，假设群体的规模不变，即遗传的过程中，每一代的群体中染色体数相同. 由于遗传算法的三个算子分别为：种群选取、交配和变异，下面三个引理分别针对这三种情况独立考虑模板的变化.

引理 4.1　在生殖过程中，若每一个个体被选入种群 newpop$(t+1)$ 的概率为(4.1)，且种群的规模与群体相同，则模板 H 所包含的染色体在 $t+1$ 时刻的期望数为

$$E_1(H,t+1) = f(H,t)N(H,t), \tag{4.3}$$

其中 $N(H,t)$ 为 t 时刻 $T(H,\text{pop}(t))$ 所包含的染色体数，

$$f(H,t) = \frac{\displaystyle\sum_{Y:Y\in T(H,\text{pop}(t))} \frac{\text{fitness}(Y)}{|T(H,\text{pop}(t))|}}{\displaystyle\sum_{Y:Y\in \text{pop}(t)} \frac{\text{fitness}(Y)}{|\text{pop}(t)|}}. \tag{4.4}$$

证明　H 模板的每一个染色体被选中的平均概率为

$$\sum_{Y:Y\in T(H,\text{pop}(t))} \frac{\text{fitness}(Y)}{|T(H,\text{pop}(t))|},$$

而群体中每一个个体被选的平均概率为

$$\sum_{Y:Y\in \text{pop}(t)} \frac{\text{fitness}(Y)}{|\text{pop}(t)|},$$

所以，(4.3) 成立. □

引理 4.2　若在 t 时刻，模板 H 的长度为 $\delta(H)$，采用简单交配方法，即随机选一个交配位，交换位后基因，交配的概率是 p_c，则在 $t+1$ 时刻模板 H 保留下来的概率为

$$p_1(H,t+1) \geqslant 1 - \frac{p_c\delta(H)}{n-1}(1-p(H,t)), \tag{4.5}$$

其中，$p(H,t)$ 表示 t 时刻模板 H 出现的概率.

证明　模板 H 发生变化的可能是由交配产生的. 由于采用交配位后的基因交换，因此选择第一个模板位到最后一个模板位中的任何一个位置为交配位都可能产生模板的变化，所以，模板发生改变的最大概率是 $p_c\frac{\delta(H)}{n-1}$. 假设交配的双方有相同的模板，则交配后模板不变. 一方不具有同 H 相同模板的概率为 $1-p(H,t)$. 于是使模板 H 改变的最大概率为

$$p_c\frac{\delta(H)}{n-1}(1-p(H,t)),$$

在交配的过程中，会出现交配的双方不具有 H 模板，但交配后却具有 H 模板，故有

$$p_1(H,t+1)\geqslant 1-p_c\frac{\delta(H)}{n-1}(1-p(H,t)).$$ □

经过简单的交配，引理 4.2 指出模板中的模板位置和元素保持不变的最小概率. 同样下面的引理 4.3 也是专门讨论模板的变化.

引理 4.3　假设模板 H 在 t 时刻存在的概率为 $p(H,t)$，经过简单变异，则有

$$p_2(H,t+1)\geqslant 1-p_m o(H), \tag{4.6}$$

其中，p_m 为每个基因变异概率，$o(H)$ 为 H 的阶.

证明　变异没有改变 H 模板的概率为

$$(1-p_m)^{o(H)}\geqslant 1-p_m o(H).$$ □

通过上面三个引理，从数学理论得到种群选取、交配和变异后，一个模板 H 的变化概率. 由此，可以综合得到下面定理.

定理 4.1　（模板定理）假设群体在 t 时刻时有相同模板 H 的染色体个数为 $N(H,t)$，经过满足引理 4.1 的种群选取、满足引理 4.2 的以概率 p_c 的交配和满足引理 4.3 以概率 p_m 的变异，则在 $t+1$ 时刻，群体中具有 H 模板的染色体数的期望值为

$$E(H,t+1) \geqslant \{1 - \frac{p_c\delta(H)}{n-1}(1-p(H,t)) - p_m o(H)\} f(H,t)N(H,t). \tag{4.7}$$

如果

$$f(H,t) \geqslant \left\{1 - \frac{p_c\delta(H)}{n-1}(1-p(H,t)) - p_m o(H)\right\}^{-1}, \tag{4.8}$$

则从概率意义来说，每代中具有 H 模板的染色体个数将随代数 t 的增加而增加.

证明 遗传算法每一代的遗传由种群选取、交配和变异三个算子组成. 三个事件的综合效果，使新一代群体中具有模板 H 的染色体数的期望值由引理 4.1、引理 4.2 和引理 4.3 得到

$$\begin{aligned} E(H,t+1) &= E(H,t+1)p_1(H,t+1)p_2(H,t+1) \\ &\geqslant \left\{1 - \frac{p_c\delta(H)}{n-1}(1-p(H,t))\right\} \\ &\quad \times \{1 - p_m o(H)\} f(H,t)N(H,t) \\ &\geqslant \{1 - \frac{p_c\delta(H)}{n-1}(1-p(H,t)) \\ &\quad - p_m o(H)\} f(H,t)N(H,t). \end{aligned}$$

□

(4.8) 的右端是一个不小于 1 的数. 由此可以推断，要使具有 H 模板的染色体个数随代数 t 增加，必须要求(4.4) 的 $f(H,t)$ 不小于 1. 直观的解释是：若要具有 Y 模板的染色体个数代代增加，必须使得它们的平均适应性不小于 1. 要使(4.7) 的右端尽量的大，可以控制的参数有：使 $\delta(H)$ 尽量小，使 $o(Y)$ 尽量小. 在平均适应性相同的前提下，新的一代群体中，具有低阶、长度短的模板的染色体期望数不低于那些具有高阶、长度长的模板的染色体期望数. 低阶、长度短且平均适应性不低于 1 的模板一般称之为"积木块"(building block).

例 4.4 用遗传算法求解 $z = \max\limits_{\substack{0\leqslant x\leqslant 31 \\ x\text{为整数}}} x^2$. 观察模板 $H1 = 1****$, $H2 = *10**$ 和 $H3 = 1***0$ 的变化情况.

表 4.3　遗传算法第一步数值结果

No.	群体	x	适应函数 $f(x)$	概率分布	$\frac{f(i)}{\sum_{i=1}^{4}\frac{f(i)}{4}}$	随机模拟选取出现次数
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.06	0.22	0
4	10011	19	361	0.31	1.23	1
合计			1170	1.00	4.00	4.0
平均			293	0.25	1.00	1.0
最大			576	0.49	1.97	2.0

模板特性(生殖前)

模板	群体	具有同模板染色体	$\sum_{i\in T(H,t)}\frac{f(i)}{\lvert T(H,t)\rvert}$	$\frac{\sum_{i\in T(H,t)} f(i)/\lvert T(H,t)\rvert}{\sum_{i\in \mathrm{pop}(t)} f(i)/\lvert \mathrm{pop}(t)\rvert}$
$H1$	1 * * * *	2,4	469	1.6
$H2$	* 10 * *	2,3	320	1.09
$H3$	1 * * * 0	2	576	1.97

按表 4.3 用轮盘赌模拟选取种群，在种群中(见表 4.4(a))11000 出现两次，01101 和 10011 各出现一次. 由引理 4.1 的(4.3)，种群中具有 $H1$ 模板的染色体期望值为

$$E_1(H1,t+1)=f(H1,t)N(H1,t)$$
$$=\frac{469}{293}\times 2\approx 3.20.$$

同样的方法可计算 $E_1(H2,t+1)=2.18$，$E_1(H3,t+1)=1.97$. 种群中具有与 $H1$，$H2$ 和 $H3$ 相同模板的染色体实际出现数(见表 4.4(b))分别为 3，2 和 2. 在选定种群后，采取两个染色体随机匹配，随机选一个位置进行位置后的基因交换，交配的概率为 1，变

异的概率为0,所得结果如表4.4(a).观察新群体中模板$H1$,$H2$和$H3$的情况,结果见表4.4(c).它们的理论期望数由(4.7)给出,计算$H1$的期望数,因为$\delta(H1)=0$,所以

$$E(H1,t+1)\geqslant f(H1,t)N(H1,t) \\ =3.20.$$

计算$H2$的期望数时,$\delta(H2)=1$, $p(H2,t)=2/4$,所以

$$E(H2,t+1)\geqslant \{1-\frac{1}{4}(1-0.5)\}f(H2,t)N(H2,t) \\ =1.90.$$

计算$H3$的期望数时,$\delta(H3)=4$,$p(H3,t)=1/4$,所以

$$E(H3,t+1)\geqslant \{1-\frac{4}{4}(1-0.25)\}f(H3,t)N(H3,t) \\ =0.49.$$

表 4.4(a) 遗传算法第二步数值结果

种群体	匹配(随机选)	交配位(随机选)	新群体	x值	$f(x)$
0110|1	2	4	01100	12	144
1100|0	1	4	11001	25	625
11|000	4	2	11011	27	729
10|011	3	2	10000	16	256
合计					1754
平均					439
最大					729

表 4.4(b) 模板特性(种群)

模 板	理论期望数	实际数	同模板染色体
$H1$	3.20	3	2,3,4
$H2$	2.18	2	2,3
$H3$	1.97	2	2,3

表 4.4(c)　模板特性(生殖后)

理论期望数	实际数	同模板染色体
3.20	3	2,3,4
1.90	2	2,3
0.49	1	4

其中,匹配是从种群中选择一个匹配的对象,如表 4.4(a) 种群中选择个体 1 和个体 2 为匹配对象,则表示他们两个成为交配的父母.表 4.4(b) 和 4.4(c) 中的同模板染色体是具有相同模板的染色体的序号.上面的数值模拟结果反映了定理 4.4 的结论.长度短的模板适应能力较强,而长度长的模板适应能力较低. □

在一个有 m 个染色体,每个染色体的编码长度为 n 的群体中,经过一代遗传可生存的模板有多少?可以这样粗略的估计:可生存的模板一定有较大的出现概率,给以一个固定的概率值 p_s,由于交配使得长度大的模板不易生存,再由引理 4.2,

$$1-\frac{p_c\delta(H)}{n-1}(1-p(H,t))\geqslant 1-\frac{p_c\delta(H)}{n-1}\geqslant p_s,$$

则要求

$$\delta(H)\leqslant\frac{(n-1)(1-p_s)}{p_c}, \tag{4.9}$$

由此只考虑长度较短的模板,考虑染色体中长度小于 δ_s 的模板.这样长度的模板数量的一个下限为 $2^{\delta_s}(n-\delta_s)$,上式的估计可以用下面的例子来解释:一个 9 个基因的染色体 * * * * * * * * *,如果仅考虑长度小于 3 的模板,则 $ $ * * * * * * *,$ 表示模板中的一个基因,* 表示不关心的基因,则将三个 $ $ 一体逐位后移至少有 $2^{3-1}(9-3+1)$ 个不同长度不超过 3 的模板.加之在群体中有 m 个染色体,故长度小于 δ_s 的模板出现数的下界为

$$m2^{\delta_s}(n-\delta_s). \tag{4.10}$$

若群体维数 $m = 2^{\frac{\delta_s}{2}}$,则长度小于 δ_s 的模板出现数为 $O(m^3)$. 这是 Goldberg[3] 估计的一个结果. 在给予一个合适的群体维数后,它隐含着遗传算法的一次进化中平行处理 $O(m^3)$ 种不同的模板. 这一性质一般被称为遗传算法的隐式并行性(implicit parallelism).

4.3 马尔可夫链收敛分析

本节所讨论的内容基于简单遗传算法.

第 3 章 3.2 节已经介绍马尔可夫链的一些基本性质. 为了进一步分析遗传算法的收敛性,再引入下面的定义和性质.

定义 4.1 一个方阵 $A \in \mathbb{R}^{n\times n}$ 称为:

(1) 非负的($A \geqslant 0$),若 $a_{ij} \geqslant 0, \forall\, i,j \in \{1,2,\cdots,n\}$ 成立;

(2) 全正的($A > 0$),若 $a_{ij} > 0, \forall\, i,j \in \{1,2,\cdots,n\}$ 成立;

(3) 本原的,如果 A 非负且存在一个整数 $k \geqslant 1$ 使得 $A^k > 0$;

(4) 归约的,若 A 非负且经过相同的行和列初等变换得到如下形式

$$\begin{pmatrix} C & 0 \\ R & T \end{pmatrix}$$

其中,C 和 T 皆为方阵;

(5) 不可约的,如果 A 非负且不是归约的;

(6) 随机的,若 A 非负且 $\sum_{j=1}^{n} a_{ij} = 1, \forall\, i \in \{1,2,\cdots,n\}$;

(7) 稳定的,若 A 是一个随机阵且所有行相同,即每一列中的元素全部相同;

(8) 列容的,若 A 是一个随机阵且每一列中至少有一个正数.

下面的引理是遗传算法收敛性分析的基础.

引理 4.4 若 C、M 和 S 是随机的,其中 $M > 0$ 和 S 是列容的,则 $CMS > 0$.

证明　记 $A = CM$ 和 $B = AS$. 因为 C 是随机的,则 C 的每一行中至少有一个正元素. 由此

$$a_{ij} = \sum_{k=1}^{n} c_{ik} m_{kj} > 0, \quad \forall\, i,j \in \{1,2,\cdots,n\},$$

即得 $A > \mathbf{0}$. 类似,因为 S 是列容的,得到

$$b_{ij} = \sum_{k=1}^{n} a_{ik} s_{kj} > 0, \quad \forall\, i,j \in \{1,2,\cdots,n\}. \qquad \square$$

引理 4.5　若 P 是本原随机阵,则 P^k 收敛到一个全正稳定随机阵

$$P^{\infty} = \lim_{k\to\infty} P^k = (1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n),$$

其中 $(p_1,p_2,\cdots,p_n)^{\mathrm{T}}$ 唯一满足,

$$(p_1,p_2,\cdots,p_n)P = (p_1,p_2,\cdots,p_n),$$

$$\sum_{i=1}^{n} p_i = 1 \text{ 和 } p_j = \lim_{k\to\infty} p_{ij}^{(k)} > 0.$$

即 $(p_1,p_2,\cdots,p_n)^{\mathrm{T}}$ 是矩阵 P^{T} 的特征值为 1 且每一个分量为正数的特征向量,且满足 $\sum_{i=1}^{n} p_i = 1$.

证明　借助定理 3.5 来证明这个结论. 感兴趣的读者可利用本章练习题 1,2,3 证明. 因为 P 为本原随机阵,将 P 看成时齐马氏链的一步转移矩阵,故有 $\forall\, i,j \in \{1,2,\cdots,n\}$,存在 k 满足 $p_{ij}^{(k)} > 0$. 由定理 3.6 知马氏链是不可约的. 再由 $P^k > 0$ 得到 $P^l > 0, l \geqslant k$,和周期性的定义,马氏链中的任何一个状态都是非周期的. 由定理 3.5 得到以下结论: 存在平稳分布 $(p_1,p_2,\cdots,p_n)^{\mathrm{T}}$ 满足

$$(p_1,p_2,\cdots,p_n)P = (p_1,p_2,\cdots,p_n),$$

$$\sum_{i=1}^{n} p_i = 1 \text{ 和 } p_j = \lim_{k\to\infty} p_{ij}^{(k)} > 0.$$

再由 $(p_{ij}^{(k)}) = P^k$(练习题 6) 推出

$$P^{\infty} = \lim_{k\to\infty} P^k = (1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n). \qquad \square$$

引理 4.6 若$P=\begin{pmatrix}C & O\\ R & T\end{pmatrix}$的随机阵，其中可约阵中$C$为一个$m\times m$全正随机阵，$R,T\neq O$则

$$P^{\infty}=\lim_{k\to\infty}P^k=\lim_{k\to\infty}\begin{pmatrix}C^k & O\\ \sum_{i=0}^{k-1}T^iRC^{k-i} & T^k\end{pmatrix}=\begin{pmatrix}C^{\infty} & O\\ R_{\infty} & O\end{pmatrix},$$

是一个稳定的随机矩阵，满足

$$P^{\infty}=\lim_{k\to\infty}P^k=(1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n),$$

$$\sum_{j=1}^{n}p_j=1 \text{ 和 } p_j=\lim_{k\to\infty}p_{ij}^{(k)}\geqslant 0,$$

其中，$p_j>0(1\leqslant j\leqslant m)$，$p_j=0(m+1\leqslant j\leqslant n)$，$R_{\infty}$表示$\sum_{i=0}^{k-1}T^iRC^{k-i}$的极限.

证明 (1) 首先证明P有一个特征值1，而其余特征值的模都小于1. 分二种情况分别讨论.

情况1 $P=(p_{ij})$的所有特征值的模都不超过1. 假设不然，存在一个特征值$\|\lambda\|>1$. 考虑特征多项式$A=\lambda I-P=(a_{ij})$，则有

$$\|a_{ii}\|-\Big(\sum_{j=1,j\neq i}^{n}\|a_{ij}\|\Big)=\|\lambda-p_{ii}\|-\sum_{j=1,j\neq i}^{n}\|p_{ij}\|$$
$$\geqslant\|\lambda\|-\|p_{ii}\|-(1-\|p_{ii}\|)$$
$$=\|\lambda\|-1.$$

因此，由练习1知A可逆，与λ是特征值矛盾.

情况2 $\|\lambda\|=1$中只有$\lambda=1$是一个特征值. 设P的一个特征值为$\lambda=e^{i\theta}$. 当$\theta\neq 2k\pi$，k为整数时，

$$\|a_{ii}\|-\Big(\sum_{j=1,j\neq i}^{n}\|a_{ij}\|\Big)=\|\cos\theta+i\sin\theta-p_{ii}\|-\sum_{j=1,j\neq i}^{n}\|p_{ij}\|$$
$$=\sqrt{p_{ii}^2-2p_{ii}\cos\theta+1}-(1-p_{ii})$$

$$= \sqrt{(1 - p_{ii})^2 + 2p_{ii}(1 - \cos\theta)} - (1 - p_{ii})$$
$$> 0.$$

同第一种情况，当 $\theta \neq 2k\pi$ 时 $\lambda = e^{i\theta}$ 不是 $\boldsymbol{P}$ 的特征值. 明显，特征值 $\lambda = 1$ 的一个特征向量为 $(1,1,\cdots,1)^{\mathrm{T}}$.

(2) 由 $\boldsymbol{P}^k = \begin{bmatrix} \boldsymbol{C}^k & \boldsymbol{O} \\ \sum_{t=0}^{k-1} \boldsymbol{T}^t \boldsymbol{R} \boldsymbol{C}^{k-t} & \boldsymbol{T}^k \end{bmatrix}$，因为 $\boldsymbol{T}$ 本身是一个方阵和 $\boldsymbol{R} \neq 0$，推出 $\boldsymbol{T}$ 中必有一行和严格小于 1，而其他行和不超过 1. 考虑特征多项式 $\lambda \boldsymbol{I} - \boldsymbol{T}$，当 $\|\lambda\| \geqslant 1$ 时，$\lambda \boldsymbol{I} - \boldsymbol{T}$ 是主对角占优，由练习 1，$\lambda \boldsymbol{I} - \boldsymbol{T}$ 可逆. 由此得到 $\|\lambda\| < 1$. 再由练习 3 得到 $\boldsymbol{T}^k \to 0$ $(k \to \infty)$.

(3) 由 $|\lambda \boldsymbol{I} - \boldsymbol{P}| = |\lambda \boldsymbol{I}_m - \boldsymbol{C}||\lambda \boldsymbol{I}_{n-m} - \boldsymbol{T}|$ 和第 2 条的结论，$|\lambda \boldsymbol{I}_{n-m} - \boldsymbol{T}| = 0$ 根的模小于 1.

设

$$\boldsymbol{C} = \boldsymbol{Q}_1^{-1} \mathrm{diag}(\boldsymbol{D}_0, \boldsymbol{D}_1, \cdots, \boldsymbol{D}_s) \boldsymbol{Q}_1,$$

其中 $\boldsymbol{D}_0$ 对应 $\lambda = 1$ 的约当标准形，$\boldsymbol{D}_1, \cdots, \boldsymbol{D}_s$ 对应特征值模小于 1 的约当标准形。因为

$$\boldsymbol{C}_k = \boldsymbol{Q}_1^{-1} \mathrm{diag}(\boldsymbol{D}_0^k, \boldsymbol{D}_1^k, \cdots, \boldsymbol{D}_s^k) \boldsymbol{Q}_1,$$

所以 $\boldsymbol{D}_i^k \to O(k \to \infty, i = 1,2,\cdots,s)$(练习 3).

因为 $\boldsymbol{C}$ 为全正矩阵，所以由引理 4.5 得 $\boldsymbol{C}^k$ 的极限存在. 由 $\boldsymbol{D}_0$ 为特征值为 1 的约当形，若 $\boldsymbol{D}_0$ 不是单位阵，则易证 $\boldsymbol{D}_0^k (k \to \infty)$ 的极限不存在，因而 $\boldsymbol{C}^k (k \to \infty)$ 的极限不存在. 由此矛盾得知 $\boldsymbol{D}_0$ 为单位阵. 由约当标准形的理论，$\boldsymbol{P}$ 在复数域可以变形为

$$\boldsymbol{P} = \boldsymbol{Q}^{-1} \mathrm{diag}(\boldsymbol{I}, \boldsymbol{J}_1, \boldsymbol{J}_2, \cdots, \boldsymbol{J}_r) \boldsymbol{Q},$$

其中，$\boldsymbol{Q}$ 是一个可逆复方阵，$\boldsymbol{I}$ 表示单位阵，$\boldsymbol{J}_i$ 表示特征值模小于 1 的约当块. 再由练习 3 和

$$\boldsymbol{P}^k = \boldsymbol{Q}^{-1} \mathrm{diag}(\boldsymbol{I}^k, \boldsymbol{J}_1^k, \boldsymbol{J}_2^k, \cdots, \boldsymbol{J}_r^k) \boldsymbol{Q},$$

得到 $\boldsymbol{P}^k(k\to\infty)$ 极限存在为 $\boldsymbol{Q}^{-1}\mathrm{diag}(\boldsymbol{I},o,o,\cdots,o)\boldsymbol{Q}$.

(4) $\lambda=1$ 的代数重数是 1. 假设 $\lambda=1$ 的代数重数大于 1,则至少存在一个与 $(1,1,\cdots,1)^{\mathrm{T}}$ 线性无关的特征向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$. 假设 $\{x_1,x_2,\cdots,x_m\}$ 中最小的元素为 $x_i(1\leqslant i\leqslant m)$. 由

$$\begin{pmatrix}\boldsymbol{C} & \boldsymbol{O}\\ \boldsymbol{R} & \boldsymbol{T}\end{pmatrix}(x_1,x_2,\cdots,x_n)^{\mathrm{T}}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$$

得到

$$c_{i1}(x_1-x_i)+c_{i2}(x_2-x_i)+\cdots+c_{im}(x_m-x_i)=0;$$

再加上 $\boldsymbol{C}$ 为全正随机阵,得到 $x_1=x_2=\cdots=x_m$.

由于同一个特征值的特征向量形成线性子空间,选 $x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$ 的第一个分量 $x_1=1$,因而 $x_1=x_2=\cdots=x_m=1$. 再由

$$\begin{aligned}&\begin{pmatrix}\boldsymbol{C} & \boldsymbol{O}\\ \boldsymbol{R} & \boldsymbol{T}\end{pmatrix}((x_1,x_2,\cdots,x_n)^{\mathrm{T}}-(1,1,\cdots,1)^{\mathrm{T}})\\ &=\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{T}((x_{m+1}\cdots,x_n)^{\mathrm{T}}-(1,1,\cdots,1)^{\mathrm{T}})\end{bmatrix}\\ &=\begin{bmatrix}\boldsymbol{0}\\ (x_{m+1}\cdots,x_n)^{\mathrm{T}}-(1,1,\cdots,1)^{\mathrm{T}}\end{bmatrix},\end{aligned}$$

得到

$$\begin{aligned}&\boldsymbol{T}((x_{m+1},\cdots,x_n)^{\mathrm{T}}-(1,1,\cdots,1)^{\mathrm{T}})\\ &=(x_{m+1},\cdots,x_n)^{\mathrm{T}}-(1,1,\cdots,1)^{\mathrm{T}},\end{aligned}$$

所以,$\boldsymbol{T}$ 有特征值 1 的特征向量,与第 2 条的结论矛盾. 因此,$\lambda=1$ 的代数重数是 1.

结合 (3) 和 (4) 的讨论,$\mathrm{rank}(\boldsymbol{Q}^{-1}\mathrm{diag}(\boldsymbol{I},0,0,\cdots,0)\boldsymbol{Q})=\mathrm{rank}\begin{bmatrix}\boldsymbol{C}^{\infty} & \boldsymbol{O}\\ \boldsymbol{R}_{\infty} & \boldsymbol{O}\end{bmatrix}=1$. 再由引理 4.5,存在唯一的 $(p_1,p_2,\cdots,p_m)$ 满足 $\boldsymbol{C}^{\infty}=(1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_m)$, $\sum_{i=1}^{m}p_i=1$ 和 $p_j>0(1\leqslant j\leqslant$

m).

综合得到，存在唯一$(p_1,p_2,\cdots,p_n)$满足，$\sum_{i=1}^{n}p_i=1,p_j=\lim_{k\to\infty}p_{ij}^{(k)}\geqslant 0$和$p_j>0(1\leqslant j\leqslant m),p_j=0(m+1\leqslant j\leqslant n)$，使得

$$P^{\infty}=\lim_{k\to\infty}P^k=(1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n).$$ □

现在研究简单遗传算法的马氏链表示方法. 记所有个体形成的空间为Ω，个体用X表示，$X\in\Omega$. 简单遗传算法是将一个维数(规模)为N的群体看成一个状态. 再记简单遗传算法的马氏链状态空间为G，简单遗传算法的状态空间维数是$M=|G|=|\Omega|^N$.

例 4.5 (续例 4.2) 由于群体选用 4 个染色体，则表 4.2 中{(0001),(0100),(0011),(1110)}为一个状态，{(0000),(0101),(0001),(0011)}也为一个状态. 染色体(个体)空间的维数是2^4，所以简单遗传算法的状态空间维数是$M=|\Omega|^N=(2^4)^4$. □

无论如何，只要染色体空间的维数有限，简单遗传算法的状态空间维数也有限并记为M. 遗传算法由 3 个算子种群选取、交配和变异组成. 由于遗传是循环往返的，为了证明容易，我们按交配、变异和种群选取顺序考虑. 交配、变异和种群选取使得一个群体(状态)变化到另一个群体(状态)，这也就是一个有限状态马氏链. 下面就按这 3 个算子分别讨论.

(1) 交配

记交配概率矩阵$\boldsymbol{C}=(c_{ij})_{M\times M}$，其中$c_{ij}$为从状态$i$经过交配变为状态$j$的概率. 一个状态经过交配运算总要变化到另一个状态，因此

$$\sum_{j=1}^{M}c_{ij}=1.$$

$\boldsymbol{C}$为一个随机矩阵.

(2) 变异

经过每个基因有相同的变异概率$p_m>0$的简单变异，两个群

体状态 i 和 j 转移的概率记为 m_{ij}，对应的概率矩阵为 $\boldsymbol{M} = (m_{ij})_{M\times M}$. 比较两个群体状态 i 和 j 中个体顺序相同和位置相同的基因，记有相同基因的位置总数记为 H_{ij}. 如两个状态

$$s_i = \{(1000),(0101),(1100)\},$$

$$s_j = \{(1010),(1110),(0101)\},$$

则 $H_{ij} = 6$. 由于采用染色体的每个基因有相同的变异概率 $p_m > 0$，则状态 i 变到状态 j 的概率为 $m_{ij} = p_m^{Nn-H_{ij}}(1-p_m)^{H_{ij}} > 0$，其中 n 为个体的编码长度，N 为群体的规模. 因此 $\boldsymbol{M}$ 是一个全正矩阵.

(3) 种群选取

简单遗传算法经过交配和变异后得到一个群体(状态)，通过种群选取变化到另一个状态. 记种群选取的转移矩阵为 $\boldsymbol{S} = (s_{ij})_{M\times M}$. 仅考虑一个状态 i 经过种群选取不变的概率 s_{ii}. 假设状态 i 由 N 个染色体 $\{X_1,X_2,\cdots,X_N\}$ 组成，则由种群选取的轮盘赌方法，一个染色体 X_j 被选中的概率为

$$\frac{\text{fitness}(X_j)}{\sum_{k=1}^{N}\text{fitness}(X_k)} > 0,$$

则状态 i 经过种群选取不变的概率

$$s_{ii} \geqslant \prod_{j=1}^{N}\left(\frac{\text{fitness}(X_j)}{\sum_{k=1}^{N}\text{fitness}(X_k)}\right) > 0.$$

所以，$\boldsymbol{S} = (s_{ij})$ 是列容的.

定义 4.2 假设优化问题要求目标函数 $f(x)$ 达到最大. 令

$$Z_t = \max\{f(X_j(t))\,|\,j = 1,2,\cdots,N\},$$

其中 $Y(t) = (X_1(t),X_2(t),\cdots,X_N(t))$ 是第 t 代遗传后得到的一个群体. 一个遗传算法收敛到全局最优，当且仅当

$$\lim_{t\to\infty}\Pr(Z_t = f^*) = 1,$$

其中，$f^* = \max\{f(X) \mid X \in \Omega\}$.

定理 4.2　若参数满足：变异概率 $0 < p_m < 1$，交配概率 $0 \leqslant p_m \leqslant 1$，则简单遗传算法不收敛到全局最优值.

证明　设状态 $\boldsymbol{Y} = (X_1, X_2, \cdots, X_N)$ 满足 $\max\{f(X_j) \mid j = 1, 2, \cdots, N\} < f^*$，$p_Y^t$ 是遗传算法第 t 代在 $\boldsymbol{Y}$ 状态的概率，则有 $\Pr\{Z_t = f^*\} \leqslant 1 - p_Y^t$. 由关于交配矩阵 $\boldsymbol{C}$、变异矩阵 $\boldsymbol{M}$ 和种群选取矩阵 $\boldsymbol{S}$ 的讨论和引理 4.4，知 $\boldsymbol{CMS}$ 为全正矩阵. 再由引理 4.5，$\lim\limits_{t\to\infty} p_Y^t = p_Y^\infty > 0$，因此，$\lim\limits_{t\to\infty}\Pr(Z_t = f^*) \leqslant 1 - p_Y^\infty < 1$.　□

定理 4.2 从概率的意义说明简单遗传算法不收敛到全局最优. 只要对简单遗传算法做一点改动，每次记录下当前最优解并将群体状态最前面增加一维存放当前最优解，则遗传算法收敛到最优解.

改进简单遗传算法的主要特征是：进化的每一代中，记录前面各代遗传的最优解并存放在群体的第一位，这个染色体只起一个记录的功能而不参与遗传运算. 此时，第 t 代进化时，马氏链状态的形式称为改进群体

$$(X_{\text{best}}(t-1), X_1(t), \cdots, X_N(t)),$$

其中 $X_{\text{best}}(t-1)$ 表示遗传到 $t-1$ 代的最优染色体. $X_{\text{best}}(t-1)$ 可以是 Ω 中的任何一个元素. 改进遗传算法的状态空间维数是 $M = |\Omega|^{N-1}$. 由于第一位的染色体不参与遗传运算，将第一位相同的改进群体归于一类. 它们具有简单遗传算法相同的交配矩阵 $\boldsymbol{C}$，变异矩阵 $\boldsymbol{M}$ 和种群选择矩阵 $\boldsymbol{S}$. 记改进简单遗传算法的交配、变异和种群选择转移阵分别为 $\boldsymbol{C}^+$、$\boldsymbol{M}^+$ 和 $\boldsymbol{S}^+$，则按第一位染色体相同的改进群体归一类有

$$\boldsymbol{C}^+ = \operatorname{diag}(\boldsymbol{C}, \boldsymbol{C}, \cdots, \boldsymbol{C}),$$
$$\boldsymbol{M}^+ = \operatorname{diag}(\boldsymbol{M}, \boldsymbol{M}, \cdots, \boldsymbol{M}),$$
$$\boldsymbol{S}^+ = \operatorname{diag}(\boldsymbol{S}, \boldsymbol{S}, \cdots, \boldsymbol{S}),$$

其中各对角阵中分别有 $|\Omega|$ 个 $\boldsymbol{C}$、$\boldsymbol{M}$ 和 $\boldsymbol{S}$.

只从第一个记忆位考虑，在一步转移中，状态 $\boldsymbol{Y}(t)=(X_{\text{best}}(t-1),X_1(t),\cdots,X_N(t))$ 到 $\boldsymbol{Y}(t+1)=(X_{\text{best}}(t),X_1(t+1),\cdots,X_N(t+1))$ 的转移概率有

$$\Pr(\boldsymbol{Y}(t+1)|\boldsymbol{Y}(t))=\begin{cases}1,\ 若\ \max\{X_1(t),\cdots,X_N(t)\}=X_{\text{best}}(t),\\ \quad (X_1(t),\cdots,X_N(t))\\ \quad =(X_1(t+1),\cdots,X_N(t+1)),\\ 0,\ 其它.\end{cases} \tag{4.11}$$

如果将第一位的染色体按目标值从好到坏的顺序排列，则第一位(4.11)的变化可以表示为一步转移概率矩阵

$$\boldsymbol{U}=\begin{pmatrix}\boldsymbol{U}_{11} & & & \\ \boldsymbol{U}_{21} & \boldsymbol{U}_{22} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{U}_{|\Omega|,1} & \boldsymbol{U}_{|\Omega|,2} & \cdots & \boldsymbol{U}_{|\Omega|,|\Omega|}\end{pmatrix},$$

其中 $\boldsymbol{U}_{ij}$ 是一个 $|\Omega|^N\times|\Omega|^N$ 矩阵，$\boldsymbol{U}$ 的每一行中恰好有一个 1，$\boldsymbol{U}_{11}$ 是一个单位矩阵.

定理 4.3 如果改进简单遗传算法按交配、变异、种群选取之后更新当前最优染色体(解)的进化循环过程，则收敛于全局最优.

证明 如果将第一位的染色体按目标值从好到坏的顺序排列，这个进化过程马氏链的一步转移概率为

$$\boldsymbol{P}^+=\boldsymbol{C}^+\boldsymbol{M}^+\boldsymbol{S}^+\boldsymbol{U}$$

$$=\begin{pmatrix}\boldsymbol{CMS} & & & \\ & \boldsymbol{CMS} & & \\ & & \ddots & \\ & & & \boldsymbol{CMS}\end{pmatrix}\begin{pmatrix}\boldsymbol{U}_{11} & & & \\ \boldsymbol{U}_{21} & \boldsymbol{U}_{22} & & \\ \vdots & \vdots & \ddots & \\ \boldsymbol{U}_{|\Omega|,1} & \boldsymbol{U}_{|\Omega|,2} & \cdots & \boldsymbol{U}_{|\Omega|,|\Omega|}\end{pmatrix}$$

$$
= \begin{bmatrix} CMSU_{11} & & & \\ CMSU_{21} & CMSU_{22} & & \\ \vdots & \vdots & \ddots & \\ CMSU_{|\Omega|,1} & CMSU_{|\Omega|,2} & \cdots & CMSU_{|\Omega|,|\Omega|} \end{bmatrix}.
$$

由 CMS 是全正推出 $CMSU_{11}$ 为全正矩阵. 由矩阵 P^+ 第一位的染色体按目标值从好到坏的顺序排列,则

$$
R = \begin{bmatrix} CMSU_{21} \\ CMSU_{31} \\ \vdots \\ CMSU_{|\Omega|,1} \end{bmatrix} \neq 0,\ T = \begin{bmatrix} CMSU_{22} & & \\ & \ddots & \\ CMSU_{|\Omega|,2} & & CMSU_{|\Omega|,\Omega|} \end{bmatrix} \neq 0.
$$

利用引理 4.6 得知,不包含最优染色体的状态在马氏链的极限分布中的概率为 0,且有包含最优染色体的状态极限分布和为 1,定理由此得证. □

定理 4.4　如果改进简单遗传算法按交配、变异后就更新当前最优染色体,之后再进行种群选取的进化循环过程,则收敛于全局最优.

证明留给读者.

本节对改进简单遗传算法从马氏链理论进行了分析. 文献[5]对编码是十进制、多种变异形式用马氏链理论分析了它们的收敛性,得到类似结果.

4.4　实现的技术问题

遗传算法同禁忌搜索和模拟退火算法一样,主要问题之一是算法如何实现的技术问题. 实现算法的各种技术问题是热门题目. 下面所讨论的一些技术方法有些是借助于直观,有些则有一定的理论. 在下面的讨论中,我们不依赖于理论地给出技术细节.

4.4.1 编码

编码是遗传算法中的基础工作之一.比较直观和常规的方法是 0,1 二进制编码,我们称这一类码为常规码.这同人类的染色体成对结构类似.这种编码方法使算法的三个算子构造比较简单.诸如前面介绍的简单遗传算法的交配、变异非常简单.对一些优化问题有其表示简单和直观的优越性.如第 1 章例 1.1 的 0-1 背包问题和例 4.1.

例 4.6 0-1 背包问题为

$$z = \max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \quad \sum_{i=1}^{n} a_i x_i \leqslant b,$$

$$x_i \in \{0,1\}.$$

问题的解是一个 0-1 向量,1 表示装包,0 表示不装.于是,可以按 $(x_1, x_2, \cdots, x_n)$ 的取值形成一个自然编码.这样的编码很直观且易于遗传算法的应用. □

采用 0-1 码可以精确地表示整数,如例 4.1 的编码.精确表示 a 到 b 整数的 0-1 编码长度 n 满足 $\frac{b-a}{2^n} < 1$,即 $n > \log_2(b-a)$.

连续变量也可以采用二进制编码,但需要考虑精度.对给定的区间 $[a,b]$,设采用二进制编码长为 n,则任何一个变量

$$x = a + a_1 \frac{b-a}{2} + a_2 \frac{b-a}{2^2} + \cdots + a_n \frac{b-a}{2^n}, \quad (4.12)$$

对应一个二进制码 $a_1 a_2 \cdots a_n$.二进制码与实际变量的最大误差为 $\frac{b-a}{2^n}$.

常规码在表示有些组合优化问题时会显得无效或不方便.如例 1.2 的 TSP 问题,它用的是图表示法.选择一个弧,分量对应的

变量为 1,否则为 0. 这样一个 n 城市 TSP 的解可用一个 $n\times(n-1)$ 的 0 1 向量表示. 虽说用常规码表示比较简单. 但是,当每两个结点有弧相连,二进制编码的所有解的个数为 $2^{n(n-1)}$. 当确定一个城市为始终点时,我们可用城市间顺序的排列来表示可行解,可行解的个数有 $(n-1)!$, 如果是对称 TSP, 可行解的个数为 $\frac{(n-1)!}{2}$. 采用这种 0-1 编码, 在计算过程中只有极小的比例 $\frac{(n-1)!}{2^{n(n-1)}}$ 为可行解,大量的计算浪费. 针对 TSP,一个自然的表示方法是 n 个城市的排列. 但由例 4.3 得知常规的交配对这种编码失效. 于是,有必要讨论这种非常规的编码方式及其交配方法.

除常规的 0-1 编码外,其他的非 0-1 码称为非常规编码. 非常规的编码同问题联系紧密,如 TSP,下面将通过例 1.5 约束机器排序问题的编码来理解非常规码.

例 4.7 约束机器排序问题为

$$\min T$$

$$\text{s.t.}\quad \sum_{t=1}^{T} x_{it} = 1,\ i = 1,2,\cdots,n,$$

$$\sum_{i=1}^{n} d_i x_{it} \leqslant c_t,\ t = 1,2,\cdots,T,$$

$$x_{it} \in \{0,1\},\ i = 1,2,\cdots,n,\ t = 1,2,\cdots,T.$$

很容易得知,它的特殊情形($c_t=c$)是装箱问题. 一个自然的编码是选用决策变量的 0-1 码,是一个 $n\times T$ 的向量. 由于 T 是决策变量,因此码的上限是不确定的. 简便的处理方法是给定一个充分大的数 K,使得在 K 个时段内一定可以达到最优解. 这样估计 K 成为问题的关键. 估计的太大会不必要的占用记忆空间,估计的太小根本达不到最优解. 即使估计 K 比较容易解决,常规的 0-1 编码还存在致命的缺点. 编码没有考虑能力的约束,对应每一个时段只有 0-1 码确定后才知道能力是否满足. 因为解空间中大量解是不

可行解.若对不满足能力约束的解采用罚值的方法,计算时间必定增加.

精确的估计是:每个解用一个 $n \times T$ 的向量表示,解空间中共有 $2^{n \times T}$ 个状态.若假设每个时段最多可以加工 p 个产品,则至少有 $2^{(n-p) \times T}-1$ 个解是不可行的,因此对这些不可行解的计算是浪费时间.当然可以动态控制 T 的大小,但是何时增加或减少 T 同样也是一个难以解决的问题.

针对这个问题,一种有效的编码方法是给产品 $(1,2,\cdots,n)$ 一个加工序 $(i_1,i_2,\cdots,i_n)$,由加工序按能力约束以最小的剩余能力依次安排时段加工,在加工的过程中不允许改变产品的加工序.如产品需求为 $(d_1=5,d_2=3,d_3=10,d_4=4)$,前5个时段可提供能力为(3,9,10,5,20),若按(1,2,3,4)顺序加工,各时段加工的产品为:第1个时段不加工,第2个时段加工产品1和2,第3个时段加工产品3,第4个时段加工产品4.很明显,最优解为第1时段加工产品2,第2时段加工产品1和4,第3时段加工产品3,最优序为(2,1,4,3). □

就这两种编码法,第1种虽然直观,但没有考虑问题的特性,在解的构造中没有考虑约束,造成计算中出现不可行解,浪费计算时间.第2种编码方法考虑问题的特性及约束情况,它的表示也非常简单,在计算中无不可行解,因此,不会因为不可行解的处理而浪费时间.同样它同TSP的城市序编码一样,存在交配和变异规则确定问题.

编码问题的讨论中包含有这样一种观点:虽然遗传算法、模拟退火、禁忌搜索等,是具有通用性的全局最优算法,如果不针对问题设计算法,其计算时间可能是非常大的;可以通过对问题的了解而换取计算时间的节省.这正是目前被一些学者接受的"无免费午餐(参考文献[6])"观点.

4.4.2　评价遗传算法的常用方法

遗传算法的模板定理只对模板的进化趋势给以讨论. 马氏链收敛分析给出概率 1 收敛的条件. 无论如何,理论分析都是无穷代进化的概率结果. 用一个方法监察计算过程中解的变化趋势,可以了解进化的程度以便决定是否继续下去. 遗传算法求解最优化问题的一个目的是得到最优解. 于是需要了解遗传算法求最优解的功效. 为了尽可能地利用信息,常用处理方法和评价法分类讨论如下.

当前最好解(best-so-far) 方法. 在每一代的进化中,记录下最好的解. 通过这个最好解展示算法的效果. 这个最好解可以用于不同算法的横向比较. 如果知道问题的下界,也可以自身比较它的功效.

在线(on-line) 比较法. 用进化过程中的每一个解来了解进化趋势,其计算公式为

$$v^{\text{on-line}}(T) = \frac{1}{T}\sum_{t=1}^{T} v(t), \tag{4.13}$$

其中,T 为当前计算中群体中出现的染色体总数,$v(t)$为第 t 个染色体的目标值.

离线(off－line)比较法. 与在线比较法类似,其计算公式为

$$v^{\text{off-line}}(T) = \frac{1}{T}\sum_{t=1}^{T} v^*(t), \tag{4.14}$$

其中,T 为当前计算中所出现的遗传代数,$v^*(t)$为第 t 代前(包括第 t 代)时所得最优目标值,即

$$v^*(t) = \max\{v^*(1), v^*(2), \cdots, v^*(t-1), v^*(\text{pop}(t))\},$$

$v^*(\text{pop}(t))$表示 pop(t)中的最优目标值. (4.13)和(4.14)的区别在于(4.13)中的 t 是记第 t 个染色体,而(4.14)的 t 是第 t 代.

(4.14)的另一种理解方法是 $v^*(t)$为第 t 代时所得的最优目

标值，即

$$v^*(t) = \max_{i \in pop(t)} \{v(i)\}. \tag{4.15}$$

(4.15)又提供一种评价进化计算的方法.如果记录当前时刻的最优解，(4.15)可以用来比较每一代中解的改进情况.

从直观可以看出，当优化问题是最大化目标函数时，随着T的增加(4.14)应该具有上升趋势.因为适应函数取决于目标函数，随着一代代进化使适应能力强的染色体生存下来，也就是目标值大的状态保留下来，(4.14)的平均值上升.

(4.13)虽说不同于(4.14)，但总的变化趋势是相同的.可能不同的地方是在每一代，(4.13)可能有波动，但代与代之间还应该保持上升趋势.

通过(4.13)或(4.14)或(4.15)的监控，可以掌握遗传算法计算的进展情况，以决定是否改进、停止算法.

4.4.3 初始参数的选取和停止原则

1. 群体的规模

在算法的第一步，需要确定群体的规模.从第4.2节最后的理论分析的结果，一个比较好的群体规模为$m=2^{\frac{\delta_s}{2}}$，其中δ_s为满足(4.9)要求的模板长度.这个群体的规模隐含$O(m^3)$个模板的并行计算.从(4.9)估计δ_s为$O(n)$，其中n为个体的编码长度，于是当编码长度增加时，群体的规模指数增加.若将这一群体规模应用在大规模的实际问题中，可以推断它们群体的规模非常之大，因此带来计算时间的增加，于是遗传算法难以同其他的算法竞争.

群体的规模取个体编码长度数的一个线性倍数是实际应用时经常采用的方法之一.如m在n和$2n$之间的一个确定数.

群体规模的选择也可以是变化的.非常直观，当(4.15)多个进化代没能改变解的性能，保持现有的群体规模已很难改进解，此时

可扩大群体的规模. 反之，若解的改进非常好，则可以减少群体的规模以便加快计算的速度.

2. 初始群体的选取

大多数学者接受这样的一个观点：初始群体应该随机选取. 只有随机选取才能达到所有状态的遍历，因而最优解在遗传算法的进化中最终得以生存. 毫无疑问，初始群体的随机选取加大了进化的代数，因而，加大了计算时间. 因此，一些学者提出应该用其他的一些启发式算法或经验选择一些比较好的染色体(种子)作为初始群体. 争论的焦点是：种子的选取有一定的偏见和缺乏代表性，因此可能产生早熟而无法求出最优解. 这个问题实际上的结论是“无免费午餐”. 应用者应针对实际问题而权衡.

3. 终止规则

一个最为简单的停止规则是给定一个最大的遗传代数 MAXGEN，算法迭代代数在达到 MAXGEN 时停止.

第 2 类方法是给定问题一个下界 LB 的计算方法，当进化中达到要求的偏差度 ε 时，算法终止，即当 $v^*(t)-\text{LB}<\varepsilon$ 时，停止.

第 3 类规则有一定的自适应性. 如(4.13)、(4.14)和(4.15)的评价算法规则，当由它们监控得到算法再进化已无法改进解的性能，此时停止计算. 如用(4.15)监控到算法已经 K 代没有进化到一个更好的解，于是算法停止.

最后一类是多种停止规则的组合. 如第 1 类和第 2 类或第 3 类准则的组合.

4.4.4　进化过程中的技术问题

这是遗传算法应用中讨论最多的内容. 主要涉及进化过程中适应函数的确定、种群的选取、交配的及变异方法等问题. 下面逐个问题讨论.

1. 适应函数

(1) 简单适应函数

简单的适应函数是目标函数的简单变形. 若 $f(x)$ 为目标函数, 则适应函数可以取

$$\text{fitness}(x) = f(x), \quad \text{优化目标为最大,} \tag{4.16}$$

$$\text{fitness}(x) = M - f(x), \quad M > c_{\max} \text{ 且优化目标为最小.} \tag{4.16'}$$

简单适应函数的优点是构造简单, 与目标函数直接相关.

值得注意的是: 采用简单适应函数可能使得算法在迭代过程中出现收敛到一些目标值近似的不同染色体, 因此再采用简单的适应函数已难区别这些染色体.

例 4.8 已知优化问题为: $z = \max\limits_{x\in[0.5,1]} f(x) = \max\limits_{x\in[0.5,1]} \{1 + \log x\}$. 若采取 (4.12) 四位编码且适应函数为优化问题的目标函数. 遗传算法进化的某一步为表 4.5:

表 4.5

x	群体	fitness(x)	概率分布 p_x
1/2	0000	0.699	0.221
5/8	0100	0.796	0.252
21/32	0101	0.817	0.259
23/32	0111	0.857	0.271

此时, 若从群体中以概率分布选取种群, 由于它们的概率分布值相差很小, 因此很难区别哪一个染色体占先. □

(2) 非线性加速适应函数

例 4.9 (续例 4.8) 从随机模拟来看, 表 4.5 的适应值改进的速度非常慢. 若将适应函数用同目标函数具有相同变化趋势, 但变化速度更快的

$$\text{fitness}(x)=\begin{cases}\dfrac{1}{1-x}, & 0.5\leqslant x<1,\\ M>0, & 1\leqslant x,\end{cases}$$

替代，其中 M 是一个充分大的数，则表 4.5 改进为表 4.5′

表 4.5′

x	群体	fitness(x)	概率分布 p_x
1/2	0000	2	0.180
5/8	0100	2.667	0.240
21/32	0101	2.909	0.261
23/32	0111	3.556	0.319

此时，已有较明显的区别. 要构造这样的替代函数需要对问题有较深入的了解，起码知道 1 是问题的最优解，这不是非常容易的. 一个有效且简便的方法是根据已有的信息构造替代函数. 如

$$\text{fitness}(x)=\begin{cases}\dfrac{1}{f_{\max}-f(x)}, & f(x)<f_{\max},\\ M>0, & f(x)=f_{\max},\end{cases}\tag{4.17}$$

其中 M 是一个充分大的数，$f_{\max}$ 是当前的最优目标值. 上式将表 4.5 改进为表 4.5″.

表 4.5″

x	群体	fitness(x)	概率分布 p_x
1/2	0000	6.329	6.329/(47.722+M)
5/8	0100	16.393	16.393/(47.722+M)
21/32	0101	25.000	25/(47.722+M)
23/32	0111	M	M/(47.722+M)

于是，(4.17) 仅根据已有的计算信息得到. 在 (4.17) 中，存在的问

题是 M 的选取，M 决定当前最优解的继承性. □

(4.17)给出适应函数的一种求解方法. 在例 4.9 适应函数构造中，M 有很重要的作用. 选取 M 的策略是：初始迭代时，M 同第一大与第二大目标差值的倒数尽量接近以避免早熟，后期迭代中逐步扩大差距. 也可以在早期迭代中用简单的适应函数，而在后期用这类加速的方法.

(3) 线性加速适应函数

对(4.17)的思想进一步系统化得到线性加速适应函数. 线性加速适应函数为

$$\text{fitness}(x) = \alpha f(x) + \beta, \tag{4.18}$$

其中 α,β 按下方程确定

$$\begin{cases} \alpha \dfrac{\sum_{i=1}^{m} f(x_i)}{m} + \beta = \dfrac{\sum_{i=1}^{m} f(x_i)}{m} \\ \alpha \max\limits_{1\leqslant i\leqslant m} \{f(x_i)\} + \beta = M \dfrac{\sum_{i=1}^{m} f(x_i)}{m} \end{cases}, \tag{4.19}$$

其中所有的 $x_i(i=1,2,\cdots,m)$为当前代群体中的染色体. (4.19)的第一个方程表示平均值变换后不变，第二个方程表示将当前最优值放大到平均值的 M 倍. 选取 M 的策略是：当目标值相差较大时，M 不要过大，以便遗传的随机性；当遗传的一个群体目标值接近时，逐步扩大 M.

由(4.19)解得

$$\alpha = \frac{(M-1)\dfrac{\sum_{i=1}^{m} f(x_i)}{m}}{\max\limits_{1\leqslant i\leqslant m}\{f(x_i)\} - \dfrac{\sum_{i=1}^{m} f(x_i)}{m}},$$

$$\beta = \frac{\sum_{i=1}^{m} f(x_i)}{m} \left[\frac{\max\limits_{1 \leqslant i \leqslant m} \{f(x_i)\} - M \frac{\sum_{i=1}^{m} f(x_i)}{m}}{\max\limits_{1 \leqslant i \leqslant m} \{f(x_i)\} - \frac{\sum_{i=1}^{m} f(x_i)}{m}} \right].$$

（4）排序适应函数

将同一代群体中的 m 个染色体按目标函数值从小到大排列，重将这些染色体按目标值由小到大记为 1 至 m. 直接取分布概率为

$$p(i) = \frac{2i}{m(m+1)}, \quad 1 \leqslant i \leqslant m, \tag{4.20}$$

这样，避开了对目标函数进行线性、非线性等加速适应函数的早熟可能，使每一代当前最好解以最大的概率 $2/(m+1)$ 遗传.

加速适应函数的思想同第 3 章模拟退火算法的接受概率思想相似，都希望开始时每一个状态有比较大的选取性，随着计算的一步步进行，逐渐拉开目标值不同所对应状态的档次. 这种相似之处可使遗传算法和模拟退火算法很好的结合. 在 4.5 节将专门介绍这个内容.

2. 交配规则

遗传算法中，交配规则较多，此处只介绍一些比较常用的规则.

（1）常用方法——双亲双子法

这种方法是在双亲确定后，以一个随机位进行位之后的所有基因对换. 对换后形成两个后代. 简单的示例如下：

```
            交配位                      交配位
              |                           |
父代 A   100|100                子代 A   100|010
父代 B   010|010   ────────→    子代 B   010|100
```

这是一种非常简单的方法. 4.2 节的理论就是在这种交配方法下得到的.

(2) 变化交配法(string-of-change crossover)

对于某些双亲,采用常规方法可能造成父代与子代完全相同,这样势必影响收敛速度和搜索范围. 如

	交配位			交配位
父代 A	1|1|0|1001	⟶	子代 A	1|1|0|1001
父代 B	1|1|0|0010		子代 B	1|1|0|0010

若选交配位在 1,2,3,则这样的交配无任何变化,因此,应该避开这三个位置. 可以采用的方法是从头开始先比较它们的相同的基因,从不同基因位置按常规方法随机选交配位. 如比较上面示例得 sssdsdd,其中 s 表示相同,d 表示不同. 交配位可以在第 4 到第 7 位随机选取.

(3) 多交配位法(multi—point crossover)

随机选择多个交配位,双亲以一个交配位到下一个交配位基因相互替代和下一个交配位到再下一个交配位不变这样交叉形成两个新的后代. 如有两个交配位

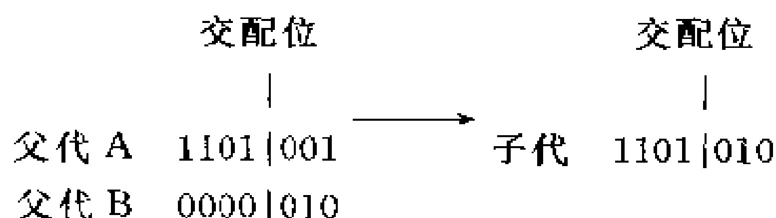
	交配位			交配位
父代 A	11|01|001	⟶	子代 A	11|00|001
父代 B	11|00|010		子代 B	11|01|010

和有三个交配位

	交配位			交配位
父代 A	11|01|00|1	⟶	子代 A	11|00|00|0
父代 B	11|00|01|0		子代 B	11|01|01|1

(4) 双亲单子法

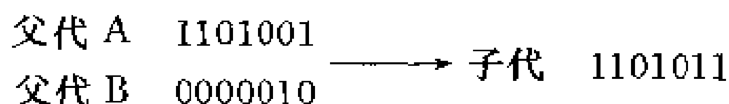
这一方法使得一对双亲只有一个后代. 一类是从常规交配法的两个后代中随机选一个,另一类则是根据优胜劣汰从两个后代中选一个好的. 如



同样,变化交配法和多交配位法也可以选择双亲单子法.

(5) 显性遗传法(dominance)

对双亲中的基因,有些是具有优超关系的,这些基因必将遗传到下一代. 例 4.1 可用来解释这种现象. 在例 4.1 中,其他位不变的情况下,任何一位的 1 永远比同位的 0 要好,于是 1 优超 0. 优超法的运算如下示例

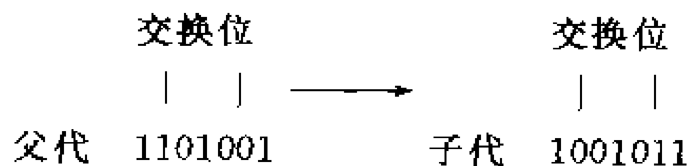


Goldberg 和 Smith[7]将这种方法成功地应用在背包问题中.

以上方法都是双亲遗传法,还有一大类为单亲遗传法.

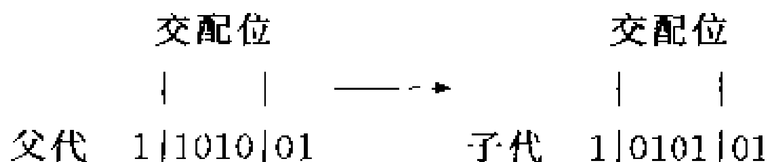
(6) 单亲遗传法

单亲遗传的一个特点是只有一个父代,下一代的产生通过单亲自身的基因变化. 如选定一个单亲,随机选两个基因位置,将两个位置的元素进行交换,见下面的交换:



也可以选定两个交配位,通过交配位之内的基因倒排得到子

代. 如



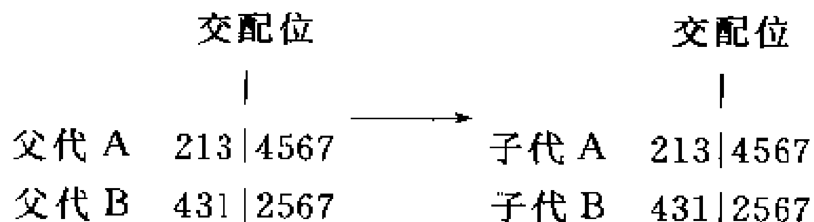
单亲遗传法使染色体中的基因取值受到限制，如上例中的 1 的个数为 4 个，0 的个数为 3 个，因此限制搜索的范围. 在使用单亲遗传时应加大变异的概率. 单亲遗传法可以同双亲遗传法结合使用.

以上讨论都局限于常规码，我们在例 4.3 和例 4.7 中讨论了非常规码，非常规码对上面讨论的一些交配方式失效，如对常规双亲双子、变化交配法、多交配位、单子法和优超法等失效. 于是，有必要讨论非常规码的交配问题.

(7) 非常规码的交配方法

此处只讨论 n 个整数 $\{1,2,\cdots,n\}$ 排列的非常规码，这种编码形式经常应用在 TSP 和约束机器排序等问题中. 一些优化问题可能有其他的非常规编码形式，它们的交配方法可根据本节介绍的思想构造.

法 1：非常规码的常规交配法. 随机选一个交配位，两个后代交配位之前的基因分别继承双亲的交配位之前基因，交配位之后的基因分别按异方基因顺序选取不重基因. 如



子代 A 是从父代 A 的交配位前取 213，然后以父代 B4312567 依顺序选不重基因 4，5，6，7.

法 2：不变位法. 随机产生一个同染色体有相等维数的不变位

向量，每一分量随机产生 0 或 1，其中 1 表示不变，0 表示变. 变化的方式按法 1 处理. 如随机产生这样的一个向量

1 0 0 1 1 0 0

则交配的变化情况为

	不变位 * * *			* * *
父代 A	2134657	⟶	子代 A	2314657
父代 B	4312567		子代 B	4132567

其中，子代 A 的形成首先将父代 A 的第 1,4,5 位不变，第 2,3 位的变化按法 1 从父代 B 按顺序取与不变位不同的基因 3,1；4,5 位不变使得 6,7 位分别选 5,7. 同样规则得子代 B.

在非常规码的交配中，变异也不能同常规码一样只是 0 或 1 的变化，这样的方法在非常规码中不可再用. 由于遗传算法是生物进化的模拟，因此，需要变异这一功能. 可以采用位置交换的方法实现这一功能. 其基本思想是例 1.8TSP 的 2-opt 法.

3. 种群的选取和交配后群体的确定

种群由适应函数所对应的概率分布以轮盘赌形式确定. 在实际模拟中所关心的模板可能出现实际选取数与引理 4.1 的理论期望数的偏差. 此时遗传算法的整体收敛性将引起偏差，于是可用监控的方法使得种群中所关心的结构在给定的范围内.

在种群的选取中，如用常用的交配方法，种群中随机选取的染色体数同群体 pop(t)的维数相等. 假设群体 pop(t)维数为 $m=2k$，则以概率分布随机选取的 m 个染色体随机结成 k 对，在交配概率为 1 的前提下生成 m 个后代.

在进化的过程中，有多个因素与群体的规模相关. 第一是种群 new-pop($t+1$)的选取，可以选择小于 pop(t)规模的种群，这样才能体现选择最优的染色体组成种群. 第二是交配过程. 交配概率不

一定为1,有些父代就不一定有子代.在采用其他的交配方式时,如双亲单子这样的方式,就无法保证有 m 个后代.于是,产生用多少个新的染色体去替代旧群体中的染色体的问题.

即使以概率1交配产生 m 个后代,则用新产生的 m 个后代(变异后)替代原有的 m 个父代会产生无法将最优解保持到下一代.如父代染色体为

(0000),(1010),(1001),(1101),

其中假设(0000)为最优解,则任何两个染色体结合都无法保证最优解遗传到下一代.基于上面的原因,种群选取的染色体个数可以不同于群体的个数;其次是交配后的子代不一定要求全部替换旧群体.由此产生一个替换问题.针对这个问题,常见的方法是:

- 种群的选取、交配和变异用常规的方法,只是在 mutpop$(t+1)$ 中选最优的 L 个染色体替换 pop(t) 中最差的 L 个染色体;
- 选择种群中染色体的个数只是群体的一个比例,此时采用常规的交配方法,交配概率为1,交配后的子代同 pop(t) 中的染色体通过筛选组成 pop$(t+1)$;
- 采用一些常用的交配方法,用交配、变异后的子代同 pop(t) 通过筛选组成 pop$(t+1)$.记替换率为

$$G(t+1) = 1 - \frac{\text{pop}(t) \cap \text{pop}(t+1)}{m}, \qquad (4.21)$$

替换率是替换问题的一个重要参数.$G=0$ 时为零替换,即 pop(t)=pop$(t+1)$,遗传一代后没有一个染色体变化.无任何进展的进化等于计算的重复.因此,我们期望替换率 $G>0$. 100%($G=1$)的替换使得新群体和旧群体的染色体相重率降低.直观看计算效果应该好.但有可能使得当前最优解无法遗传下去.由4.3节的理论,我们希望当前最优解保存下去.

低替换率产生过多地重复计算适应函数值,使得搜索的范围

扩展较慢.但优点是使某些关心的染色体得以保留. Whitley[8]通过计算模拟说明替换率的大小与评价函数的关系,100%替换率并不是最好的.

在群体或种群中有时会出现相同的染色体.相同的染色体造成适应函数的重复计算,但同时也是适应能力的一种表现,有扩大下一代相同染色体的生存可能性.对这个问题不必作太多的处理.

100%的替换会出现当前最优解的遗失,一种保持的策略是使上一代的当前最优解强行遗传到下一代.

本节主要介绍遗传算法中应注意的问题和各种处理方法.我们没有讨论它们之间的关系和实际应用的最佳搭配.我们坚持"无免费午餐"的观点,参考文献[9].因此用遗传算法求解优化问题时,应尽可能地了解问题的本身结构,针对问题给出算法设计,决对不能无目的的模拟计算.

4.5 遗传模拟退火算法

第 4.1 节中提到遗传算法的兼容性,本节简单介绍模拟退火算法同遗传算法的一种结合算法.在模拟退火时齐算法中,每两个温度之间的状态点是无关的.从理论上来看,任何一个温度马氏链都逐渐达到平稳分布,即从一个状态到达另一个状态随着迭代的步数增加渐渐不依赖起点状态且在每一个状态的概率服从平稳分布.遗传算法强调的是两代之间的进化关系,但其交配有可能使最好解遗失.结合这两个算法的特点,构成下面的遗传模拟退火算法.

遗传模拟退火算法

STEP1 给定群体规模 maxpop, $k:=0$; 初始温度 $t_k:=t_0$,群体 pop(k);

STEP2 若满足停止规则，停止计算；否则，在群体 pop(k)中每一个染色体 $i \in \text{pop}(k)$的邻域中随机选一状态 $j \in N(i)$，按模拟退火中的接受概率

$$A_{ij}(t_k) = \min\left\{1, \exp\left(-\frac{f(j) - f(i)}{t_k}\right)\right\}, \qquad (4.22)$$

接受或拒绝 j，其中 $f(i)$为状态 i 的目标值；这一阶段共需 maxpop 次迭代选出新群体 newpop1($k+1$)；

STEP3 在 new-pop1($k+1$)中计算适应函数

$$f_i(t_k) = \exp\left\{-\frac{f(i) - f_{\min}}{t_k}\right\}, \qquad (4.23)$$

其中，$f_{\min}$是 newpop1($k+1$)中的最小值；由适应函数决定的概率分布从 newpop1($k+1$)中随机选 maxpop 个染色体形成种群 newpop2($k+1$)；

STEP4 按遗传算法的常规方法进行交配得到 crosspop($k+1$)；再变异得到 mutpop($k+1$)；

STEP5 $t_{k+1} := d(t_k), k := k+1, \text{pop}(k) = \text{mutpop}(k)$，返回 STEP2.

遗传模拟退火算法中，STEP2 的群体选择较遗传算法的选择范围要大，用 $\sum_{i \in \text{pop}(k)} N(i)$取代遗传算法中的 pop($k$). 但它并不是简单地随机选取，而是采用(4.22). 这里具有模拟退火的第一个特征. 适应函数的加速特性已在 4.4.4 节讨论，(4.23)是一个非常好的加速适应函数. 当温度较高时加速性不明显，当温度较低时加速性非常明显. 这正是我们需要的，也是模拟退火的第二个特征. 其他计算步骤同遗传算法.

4.6　应用案例——生产批量问题

本节介绍遗传算法在生产批量问题中的应用.生产批量问题是企业生产和管理中一个常见的问题.主要考虑最优的生产批量使得生产费用、生产准备费用和库存费用综合指标最小.按传统将企业内部的生产管理按图 4.3 分为 3 个层次.第 1 个层次为企业的发展战略和长期规划,是一个企业为了生存和参加市场竞争对企业发展所规划的一个方向.这个层次是战略性的.第 2 个层次是企业中、短期计划.主要体现为根据人力、物力的能力对产品需求,包括订货、计划库存等.第 3 个层次是生产计划的具体实现是对已安排的生产计划具体落实到人和机器.最下一层的具体安排可能发现能力的不足,此时,可采取一些补救的方法,如人力不足可以加班加点弥补,在最下层内部解决;也可以反馈到上层对生产计划甚至发展规划进行更改.

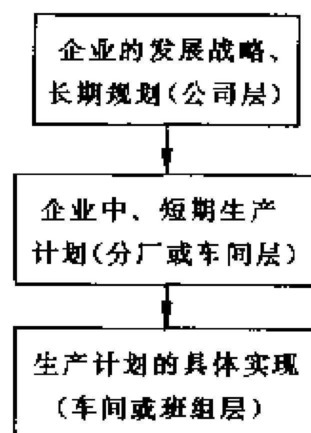


图　4.3

物料需求计划(MRP,material requirement planning)是在不考虑能力约束的前提下,根据产品需求按生产的工艺流程 BOM (bill of materials)进行物料的计划管理.它是利用主生产计划(由第 2 层决定)来决定需要哪些材料及需要哪些零部件来完成生产计划.MRPII(manufacturing resources planning)是在 MRP 的基础上于 70 年代发展起来的,它主要在考虑了市场的需求、企业的生产能力和对用户的服务水平等诸多因素的前提下来安排企业的生产.

在 MRPII 中,第 2 层的主要问题之一是生产计划中的约束批

量问题[10]. 在满足能力约束和生产平衡的条件下,能力约束批量问题是考虑生产费用、库存费用和生产准备(set-up)费用综合目标的优化问题. 它的数学模型为

$$\text{mincost} = \min \sum_{i=1}^{n} \sum_{t=1}^{T} \{s_i Y_{it} + c_i x_{it} + h_i I_{it}\} \tag{4.24}$$

$$\text{s.t. } I_{i,t-1} + x_{it} - I_{it} = d_{it} + \sum_{j \in S(i)} r_{ij} x_{jt}, \quad i = 1,2,\cdots,n; \quad t = 1,2,\cdots,T. \tag{4.25}$$

$$\sum_{i=1}^{n} (a_{kit} x_{it} + A_{kit} Y_{it}) \leqslant c_{kt}, \quad k = 1,2,\cdots,K; \quad t = 1,2,\cdots,T. \tag{4.26}$$

$$Y_{it} = \begin{cases} 1, & x_{it} > 0, \\ 0, & x_{it} = 0, \end{cases} \quad i = 1,2,\cdots,n; \; t = 1,2,\cdots,T. \tag{4.27}$$

$$I_{i0} = I_{iT} = 0,\; I_{it} \geqslant 0,\; x_{it} \geqslant 0,\; i = 1,2,\cdots,n; \; t = 1,2,\cdots,T. \tag{4.28}$$

其中

T: 计划时段,如一年的 12 个月,一周 5 天等;n: 产品的品种数;x_{it}: 为产品 i 在时段 t 的生产批量,是决策变量;I_{it}: 为产品 i 在时段 t 的库存量,是决策变量;d_{it}为时段 t 时产品 i 的需求量;s_i为生产产品 i 时的生产准备费用,这一费用是由于机器更换加工产品而一次性发生的;c_i为单位产品 i 的生产费用;h_i为单位产品 i 的库存费用;a_{kit}为 t 时段单位产品 i 占用资源 k 的量;A_{kit}为 t 时段产品 i 的生产准备占用资源 k 的量;c_{kt}为 t 时段资源 k 的可提供量;r_{ij}为 j 产品的生产过程中对 i 产品的需求量,如一个部件直接需要若干个零件,称 j 为 i 的直接后继;$S(i)$为所有 i 的直接后继集合.

在上面的能力约束批量模型中,生产准备费用、生产费用和库

存费用只与产品有关而与时段无关,可以更为复杂地考虑与时段有关的模型.从下面的介绍中可以看出,费用与时段有关模型的难度更大.

(4.24)为目标函数,它要求生产准备、库存、生产和加班费用的综合最小.方程(4.25)是物流的平衡方程,当前时段生产的量和上时段的库存量应该满足外部和内部后继产品的需求及本时段库存的需要.(4.26)是能力约束方程.(4.27)标记产品 i 在时段 t 是否生产,1 表示生产.(4.28)表示生产批量不是负数和不允许有欠货.

生产中常用的生产结构有:串联型(图 4.4a)、组装型(图 4.4b)和一般型(图 4.4c).

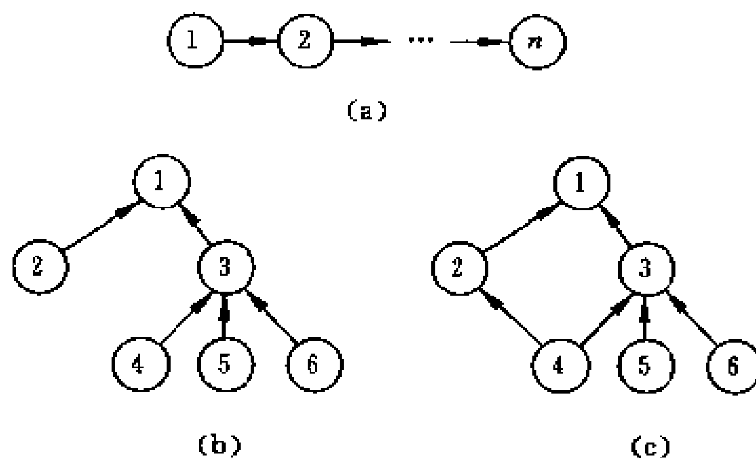


图 4.4　(a) 串联型　(b) 组装型　(c) 一般型

一个产品按以下工序进行生产:原材料——成型——精加工,是一个典型的串联型生产.组装型生产以汽车制造、飞机制造业为典型,是由零件、部件到成品的一个生产过程.一般型生产包含前两种并比它们更为广泛.在多数企业生产中都会遇到这样的情况,比较典型的是化工企业,投入几种原材料会产生出几种产品.如炼油过程,从原油变成汽油、柴油和煤油等.

产品的生产结构主要体现在(4.25)的内部需求关系 r_{ij}. 约束批量问题中,当约束(4.26)的 A_{kit} 不全为零,求解(4.25)到(4.28)的一个可行解是 NP-Complete[11]. 去掉约束(4.26)的多层无约束批量问题仍然是 NP-hard[12]. 于是,我们尝试用遗传算法求解.

1. 编码

遇到的第一个问题是编码. 由于约束批量问题是一个混合整数规划问题,决策变量 I_{it},x_{it} 为连续变量,Y_{it} 为 0-1 整数变量. 于是需要研究变量之间的关系和性质. 首先研究的是无能力约束批量问题:去掉能力约束(4.26).

定理 4.5 无能力约束批量问题(4.24)、(4.25)、(4.27)和(4.28)有满足下面条件的最优解

$$x_{it}I_{i,t-1} = 0. \tag{4.29}$$

证明 按产品生产的阶段性,由(4.25)和(4.28),产品 i 的生产量为一个常数,即有

$$\begin{aligned}\sum_{t=1}^{T}x_{it} &= \sum_{t=1}^{T}(I_{i,t-1} + x_{it} - I_{it}) = \sum_{t=1}^{T}\Big(d_{it} + \sum_{j\in S(i)} r_{ij}x_{jt}\Big) \\ &= \sum_{t=1}^{T}d_{it} + \sum_{j\in S(i)} r_{ij}\sum_{t=1}^{T}x_{jt},\ i = 1,2,\cdots,n.\end{aligned}$$

如在生产的最后一个阶段,假设 $i=1$ 为最终产品,则

$$\sum_{t=1}^{T}x_{1t} = \sum_{t=1}^{T}d_{1t};$$

在倒数第二个阶段,假设产品标号为 $i=2$ 且为第一个产品提供中间产品,则

$$\sum_{t=1}^{T}x_{2t} = \sum_{t=1}^{T}(d_{2t} + r_{21}x_{1t}) = \sum_{t=1}^{T}d_{2t} + r_{21}\sum_{t=1}^{T}d_{1t}.$$

再由(4.24)知三项费用中的生产费用 $\sum_{i=1}^{n}\sum_{t=1}^{T}c_ix_{it} = \sum_{i=1}^{n}c_i\sum_{t=1}^{T}x_{it}$ 为常数. 因此,只需讨论生产准备费用和库存费用.

假设无能力约束批量问题有一个最优解满足 $x_{it}I_{it-1}>0$，表示在 $t-1$ 时段末有库存而第 t 时段还要生产，则可以重新安排生产：在 t 时段之前新的批量 x_{it}^* 满足

$$x_{it}^* = x_{it} + I_{it-1},\ \sum_{s=1}^{t-1} x_{is}^* = \sum_{s=1}^{t-1} x_{is} - I_{it-1}.$$

在之后保持原来批量不变. 新的批量满足(4.25)、(4.27)和(4.28)且库存费用不增、生产准备费不增和生产费不变，总体目标值不增，但此时 $x_{it}I_{it-1}=0$. 由此证明定理成立. □

由定理 4.4，产品的生产满足：

(1) 当 $Y_{it}=0$ 时 $x_{it}=0$；

(2) 当 $Y_{it_1}=1, Y_{it_2}=1, 1\leqslant t_1<t_2\leqslant T$ 且 $Y_{it}=0, t_1<t<t_2$ 时，则

$$x_{it_1} = \sum_{t=t_1}^{t_2-1}\Big(d_{it} + \sum_{j\in S(i)} r_{ij}x_{jt}\Big). \tag{4.30}$$

上面的讨论给出 0-1 变量 Y_{it}同生产批量的关系. 根据 Y_{it}任何一个时段要么不生产，要么按(4.30)生产足以满足下一个生产准备时段前的所有产品需求. 对无能力约束批量问题，可以用 Y_{it}对应的 0-1 编码采用遗传算法求解. 对有能力约束的批量问题如何采用上面已经得到的结果？

处理的方法之一是针对能力可以通过其他方式，如加班或租借等. 此时可以去掉能力约束(4.26)，用一个第 t 时段第 k 种资源的加班量 O_{kt}去松弛(4.26)的约束，但在目标(4.24)中体现.

$$\text{mincost}=\min\sum_{i=1}^{n}\sum_{t=1}^{T}\{s_iY_{it}+c_ix_{it}+h_iI_{it}\}+\sum_{k=1}^{K}\sum_{t=1}^{T}\rho_kO_{kt}, \tag{4.24'}$$

其中

$$O_{kt} = \max\Big\{\sum_{i=1}^{n}(a_{kit}x_{it} + A_{kit}Y_{it}) - c_{kt}, 0\Big\},$$
$$k = 1,2,\cdots,K;\ t = 1,2,\cdots,T,$$

ρ_k为增加单位资源 k 的费用. 这样的处理方法有其应用背景，一些

企业经常在能力不足的情况下通过租借或是加班加点扩大生产能力.由于增加 O_{it},能力约束已转移到目标函数中考虑,于是无能力约束批量问题的编码可以继续使用.

另一种处理方法是针对能力不可改变而设计.它还是利用无能力约束批量问题的结果,对给定的 0-1 编码 Y_{it},按(1)和(2)进行批量生产.可能产生能力不满足要求,此时用批量模型中经常使用的"移动"启发式方法[13].移动启发式方法的基本思想是:首先按(1)和(2)确定每个时段的生产批量.第二,从最后一个时段 T 到第一个时段逆顺序检查是否满足能力约束.满足约束的条件下继续向前检查.不满足能力约束的条件下,将该时段的加工多余部分前移.当然这里面存在前移到哪一个时段的技术问题.如考虑将超出能力的生产量按最近时段有能力空闲前移.第三,为目标值的计算.若将所有不满足能力约束的时段都前移至可行,此时虽说按(4.27)$Y=\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$的生产批量同移动后的生产批量不同,但我们还是将移动后的可行解对应原有的编码 $\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$,由此按移动后的解求出目标值作为 $\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$ 的一个评价值,记为 $v(Y)$.若无法将所有不满足能力约束的时段都前移至可行,此时认为无可行解,目标值定义为一个充分大的数.

2. 适应函数

由于有能力约束批量问题的目标是极小目标函数,适应函数与群体各目标值的关系是:给定一个群体 $\{Y^1,Y^2,\cdots,Y^{maxpop}\}$、常数 $\varepsilon>0$ 和一个 0-1 码 $Y=\{Y_{it}|i=1,2,\cdots,n;t=1,2,\cdots,T\}$,适应函数为

$$\text{fitness}(Y)=\max_{1\leq i\leq \text{maxpop}} v(Y^i)-v(Y)+\varepsilon, \qquad (4.31)$$

其中,maxpop 表示群体的维数在可以加班模型和不可加班模型中 $v(Y)$分别对应各自的目标值和评价值.(4.31)中的 ε 越小,具有最大目标值(评价值)的状态被选取为种群的概率越小.

3. 实现的参数选取及计算结果

由上面的讨论，每一个解可以编码为

$$Y^{g,j}=(Y_{11}^{g,j},Y_{12}^{g,j},\cdots,Y_{1T}^{g,j},Y_{21}^{g,j},Y_{22}^{g,j},\\ \cdots,Y_{2T}^{g,j},\cdots,Y_{n1}^{g,j},Y_{n2}^{g,j},\cdots,Y_{nT}^{g,j}),$$

$$j=1,2,\cdots,\mathrm{maxpop};\ g=1,2,\cdots,\mathrm{maxgen},$$

其中 maxgen 为一个给定整数，表示算法迭代的代数. 在解的编码中为避免不可行解，假设

$$d_{i1}>0,\ Y_{11}^{g,j}=Y_{21}^{g,j}=\cdots=Y_{n1}^{g,j}=1.$$

批量由(1)和(2)确定. 在能力不可以加班时，再用“移动”的方法修正批量.

根据(4.31)计算适应函数，用概率分布

$$\mathrm{fitness}(Y^{g,j})=\frac{\mathrm{fitness}(Y^{g,j})}{\sum_{i=1}^{\mathrm{maxpop}}\mathrm{fitness}(Y^{g,i})},$$

选种群. 种群的规模同群体的规模相同.

采用双亲双子交配法，交配概率 $p_c=0.6$. 采用交配后代直接替代双亲，而没有交配的双亲直接遗传到下一代，也就是交配后群体规模同旧群体规模相同，覆盖率为 60%. 再以 $p_m=0.033$ 的概率变异.

对 $n=7,T=6,K=1$ 用十组随机模拟数据比较模拟退火、禁忌搜索和遗传算法在有能力约束、无加班和组装型生产模型的应用效果，发现计算时间和结果相差不是很大. 在可加班、一般型生产模型中，对一个 $N=21,T=6,K=2$ 的实际问题进行求解. 在主频为 66Hz 的 PC-486 上用近十分钟得到满意解.

练 习 题

1. 方阵 A 主对角占优的定义为：

$$\| a_{ii} | \geqslant \sum_{j=1,j\neq i}^{n} \| a_{ij} \| ,$$

且其中至少有一个不等式成立，其中 $\| x \|$ 表示 x 的模. 证明：主对角阵占优的方阵可逆.

2. 若 A 为随机矩阵，则 A 除有一个值为 1 的特征根，其他特征值的模均小于 1. 当 A 为全正的随机矩阵时，特征值 1 的代数重数是 1.

3. 对约当矩阵 $A=\begin{pmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ 0 & 0 & \lambda & \cdots & 0 \\ & \cdots & & \cdots & \\ 0 & 0 & 0 & \cdots & \lambda \end{pmatrix}$，当 $\| \lambda \| <1$ 时，证明：$\lim\limits_{k\to\infty} A^k=0$.

4. 若 P 为全正随机矩阵，则 P^k 收敛到一个全正稳定随机阵

$$P^{\infty} = (1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n),$$

其中

$$(p_1,p_2,\cdots,p_n)P = (p_1,p_2,\cdots,p_n) .$$

5. 若 A 为本原随机矩阵，则有 P^k 收敛到一个全正稳定随机阵

$$P^{\infty} = (1,1,\cdots,1)^{\mathrm{T}}(p_1,p_2,\cdots,p_n),$$

其中

$$(p_1,p_2,\cdots,p_n)P = (p_1,p_2,\cdots,p_n) .$$

6. 若 P 为时齐马氏链的一步转移矩阵，证明：k 步转移矩阵为 $(p_{ij}^{(k)})=P^k$.

7. 证明定理 4.4.

8. 对感兴趣的组合优化问题，比较禁忌搜索、模拟退火和遗传算法的计算效果.

参 考 文 献

1. Holland J H. Adaptation in Natural and Artificial Systems. MIT Press, 1975
2. 徐维衡主编. 医学遗传学基础. 北京医科大学、中国协和医科大学联合出版社, 1993
3. Goldberg D E. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, 1989
4. Rudolph G. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Networks, 1994, 5:96～101
5. 余春峰. 十进制遗传算法的理论分析及其应用. 清华大学硕士论文, 1998
6. Wolpert D H, Macready W G. No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995
7. Goldberg D E, Smith R E. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987. 59～68
8. Whitley D. 1987, Using reproductive evaluation to improve genetic search and heuristic discovery. In: Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, 1987. 108～115
9. Xie J, Xing W. Incorporating domain-specific knowledge into evolutionary algorithms. Beijing Mathematics, 1998, 4:131～139
10. 谢金星. 生产计划和调度:数学模型及算法. 清华大学博士论文, 1995
11. Maes J, McClain J O, Van Wassenhove L N. Multilevel capacitated lotsizing complexity and LP-based heuristics. European Journal of Operational Research, 1991, 53:131～148
12. 谢金星, 姜启源, 邢文训 et al. 能力受限的批量问题的数学模型与算法

新进展. 运筹学杂志,1996, 15(1), 1～12

13. Clark R, Armentano V A. A heuristic for a resource-capacitated multi-stage lot sizing problem with lead-time. Journal of Operational Research Society, 1995, 46:1208～1222

第 5 章

人工神经网络

神经网络是大脑的一个组成部分. 神经网络的研究可以追朔到 19 世纪. James[1] 在 1890 的《心理学(Psychology (Briefer Course))》一书中,这样描述神经网络的基本原理:

大脑皮层每一点的活力是由其他点势能释放的综合效能产生. 这一势能同下面的因素有关:(1) 相关其他点的兴奋次数;(2) 兴奋的强度;(3) 与其不相连的其他点所接受的能量.

他的这一原理一直沿用至今. 神经网络的研究分为两派. 一派主要包括生物学家、物理学家和心理学家. 他们研究的主要目的是给出大脑活动的精细模型和描述. 另一派主要包括工程技术人员. 他们关心的是怎样利用神经网络的基本原理,来构造解决实际问题的算法,使得这些算法具有有趣和有效的计算能力. 本章内容属于后者. 我们称后者为神经网络的工程应用研究,或称为人工神经网络.

神经网络的基本原理是构造人工神经网络模型的一个基本依据. 我们在后面的各节中所介绍的模型就是基于这样的原理. 本章介绍人工神经网络(artificial neural networks)的两类主要模型:前向神经网络和反馈神经网络. 主要内容包括模型、基本原理和应用.

人工神经网络的早期工作可以追朔到 1943 年 McCulloch 和 Pitts[2] 建立的第一个模型,后被扩展为"认知(perceptron)"模型. 认知模型的一个功效可以用来解决简单的分类问题. 1969 年 Minsky 和 Papert[3] 在《认知论(Perceptrons)》一书指出:认知模型

无法解决最为经典的“异或(XOR——exclusive-or)”问题.这个结论使得人工神经网络陷入一次危机.产生这次危机的原因可以归结为以下主要原因:(1) 人们对人工神经网络的认识不足和认识上的错误,(2) 人们过高的期望和夸大,(3) 宣传中的错误引导.实际上,Minsky 和 Papert 主要研究的是单隐含层的认知网络模型,他们证明了简单的线性感知功能是有限的.这个结论不应该对人工神经网络进行全面的否定.

20 世纪 80 年代,Hopfield[4,5]将人工神经网络成功地应用在组合优化问题,McClelland 和 Rumelhart[6]构造的多层反馈学习算法成功地解决了单隐含层认知网络的“异或”问题及其他的识别问题.他们的突破使得人工神经网络成为新的研究热点.

对人工神经网络的总体评价可以归为以下若干方面:

- 提供信息处理的一种新的手段.主要归功于人工神经网络的自适应性、学习能力和大规模的平行计算能力;
- 发展已较为成熟.成熟的三个方面是:(1) 模型的建立和数学理论的支持,(2) 计算工具的发展,(3) 神经生物学的发展和对大脑的认识;
- 人工神经网络发展仍然受到限制.尽管近 20 年计算机发展迅速,同时有大量的数学理论支持和发展,但计算机设备在储存、速度和用户柔性方面的非适应能力,限制了人工神经网络的发展;
- 有成功应用的案例.人工神经网络在视觉、语言、信号处理和机器人等方面有成功应用的示例.虽说应用的范围有限,但其应用的问题及效果给人们的印象深刻.

人工神经网络的模型要求发展神经网络型计算系统来替代传统的计算机.这种计算系统不再是传统计算机顺序执行命令的运行过程,而是希望对输入进行平行处理;这种计算系统不再是只包含一个或几个复杂的计算设备,而是由众多简单设备有机组成在

一起；这种计算系统处理信息时，不再是将信息存储在一个精确的位置上，而是通过神经元的内部相连关系达到信息储存的功能.

在目前的条件下，还只能依赖传统的计算机来模拟人工神经网络的这些功能. 数值计算和分析是模拟中的一个重要部分. 只有当人工神经网络硬件系统得到发展，才能使目前的人工神经网络更快、更有效地解决更大的实际问题.

5.1　人工神经网络的基本概念

人工神经网络模型是基于生物学中的神经网络的基本原理而建立的. 图 5.1 是生物学中神经网络的简图.

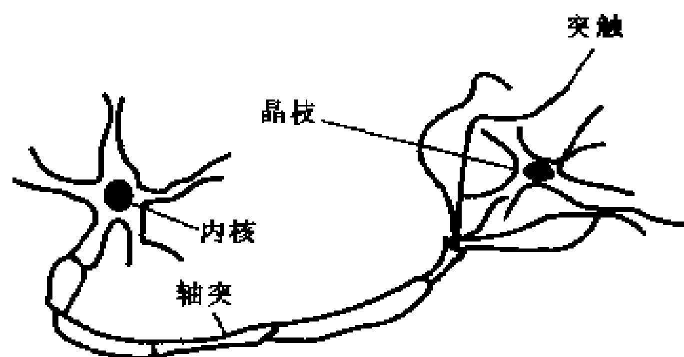


图 5.1　生物学中神经网络简图

大脑的一个重要成分是神经网络. 神经网络由相互关联的神经元组成. 每一个神经元由内核(body)、轴突(axon)和多个晶枝(dendrite)组成. 晶枝形成一个非常精密的"毛刷"环绕在内核周围. 轴突可以想象为一根又长又细的管道，其终点分为众多细小分支，将内核的信息传递给其他内核的晶枝. 这些细小分支的头，即那些又长又细管道的终点，称为突触(synapse)，它们的主要功能是接触其他内核的晶枝.

一个神经元通过晶枝接收到一定的信息后，它对这些信息进

行处理,再通过它所控制的突触传送给其他的神经元.神经元可分为"抑制"性的或"兴奋"性的两种.当一个神经元的晶枝接收的兴奋性信息累计超出某一值时,这个固定值称为阈值(threshold),这个神经元被激活并传递出一个信息给其他神经元.这种传递的信息的神经元为"兴奋"性的.第二种情况是神经元虽然接收到其他神经元传递的信息,但没有向外传递信息,此时,称这个神经元为"抑制"性的.

简单模拟上面原理的人工神经网络是 McCulloch-Pitts 认知网络.假设一个神经元通过晶枝接收到 n 个信息,McCulloch-Pitts 认知网络由图 5.2 表示.

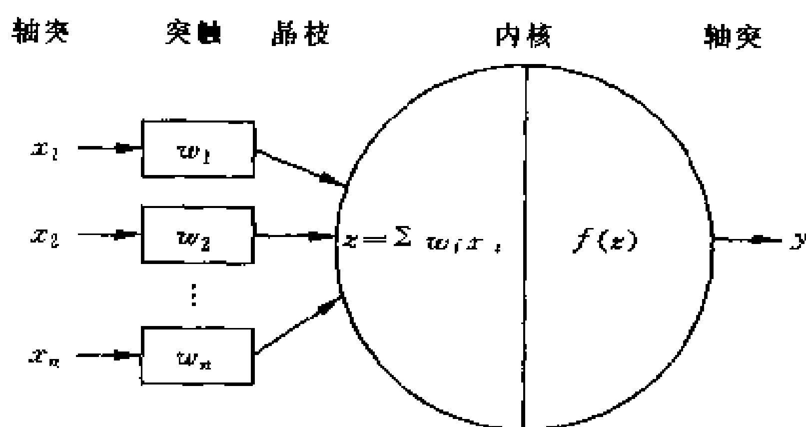


图 5.2 McCulloch-Pitts 网络

在图 5.2 中,w_i 为关联权,表示神经元对第 i 个晶枝接收到信息的感知能力.函数 $f(z)$ 称为输出函数或激活函数(activation function).采用激活函数的人工神经网络也称阈网络.McClulloch-Pitts 输出函数定义为

$$y = f(z) = \operatorname{sgn}\left(\sum_{i=1}^{n} w_i x_i - \theta\right), \tag{5.1}$$

其中,

$$\operatorname{sgn}(x)=\begin{cases}1, & x>0,\\ 0, & 其他,\end{cases}$$

θ 称为阈值.

从方程(5.1)可以看出,当 w_i 为固定值时,对给定的一组输入 $(x_1,x_2,\cdots,x_n)^T$,很容易计算得到输出值.我们的想法就是对给定的输入,尽可能使(5.1)的计算输出同实际值吻合.这就要求确定参数 w_i.人工神经网络的主要工作就是建立模型和确定 w_i 的值.依据人工神经网络模型的建立和权数 w_i 的确定方法,本章将分别介绍神经网络中的两个主要模型:前向(feed-forward)和反馈(feed-back)型人工神经网络.

前向型人工神经网络的特点是将神经元分为层,每一层的神经元之间没有信息交流,并且按一层一层地且同步计算.反馈型神经网络则将整个网络看成一个整体,神经元相互作用,计算是整体性的.

在已知一组正确的输入输出结果的条件下,人工神经网络通过这些数据,调整并确定权数 w_i 的方法称为有指导学习.在只有输入数据而不知输出结果的前提下,确定权数的方法称无监督学习.

对前向型人工神经网络,一般将人工神经网络的计算分为两个阶段:学习阶段和应用或称工作阶段.第一个阶段为学习阶段,这个阶段的主要工作是确定权数 $w_i(i=1,2,\cdots,n)$.第二个阶段是应用或工作阶段,是在权数确定的基础上,用带有确定权数的神经网络去解决实际问题.当然,学习和应用并不是绝对地分为两个阶段,他们相互相承.可以通过学习、应用、再学习、再应用的循环过程,逐渐提高人工神经网络的应用效果.

图 5.3 是前向型人工神经网络的计算流程.第一个阶段如图 5.3(a)描述,它的主要工作是:在选择网络模型和学习规则后,根据已知的输入和输出学习数据,输入和输出学习数据也分别称为

理想输入和输出数据，通过学习规则确定神经网络的权数. 这个阶段的输入是学习数据中的输入数据，通过人工神经网络的输出同学习数据中的理想输出数据的比较，确定神经网络中的权数. 尤如一个医学院的学生，通过教科书中病例的发病症状和诊断结果，来学习诊断. 第二个阶段如图 5.3(b)描述，它的主要工作是：根据第一个阶段确定的模型和得到的权数，在输入实际问题的输入数据后，给出一个结论. 尤如一个医学院的毕业生，在遇到病人后，根据医学院学到的诊断方法，给病人一个诊断.

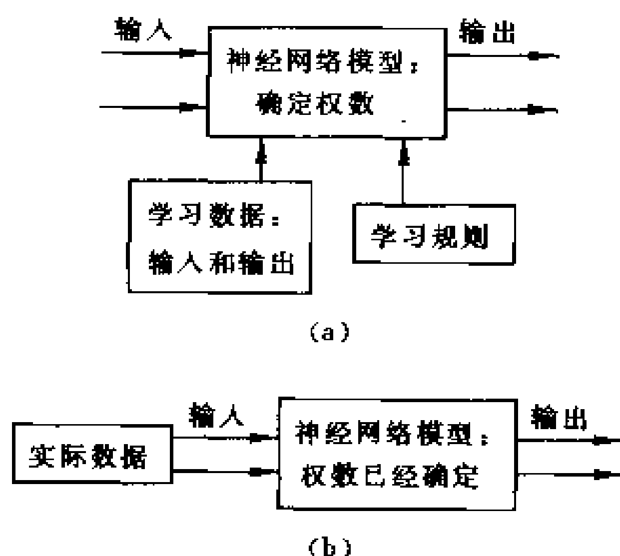


图 5.3 人工神经网络计算过程示意图

依据它的层数和激活函数的类型，前向型神经网络还可以进一步细分. 下面就从简到难的情况来逐一介绍前向型神经网络.

5.2 单层前向神经网络

单层前向神经网络(single layer feed-forward neural networks)的模型结构如图 5.4.

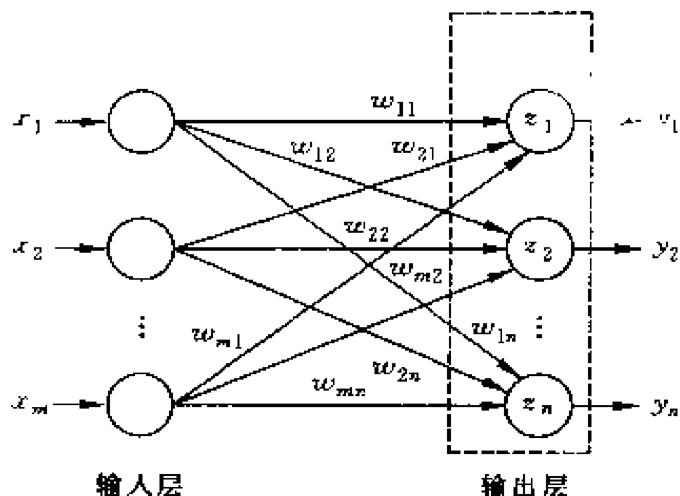


图 5.4　单层前向神经网络

在单层前向神经网络中，只有输入层和输出层. 输入层中的每一个元素表示轴突，而输出层中的每一个元素代表一个神经元的内核，由此得名为单层神经网络. 它的特点是只有一个输入层和一个输出层. 单层前向神经网络中各层中的元素无权数相连. 输出层通过对输入层的输入数据和同理想输出结果的比较，确定其权数. 最后，用确定了权数的神经网络应用到实际问题中. 这类神经网络主要在识别问题中有较强的能力.

若记输入变量为 $\boldsymbol{X}=(x_1,x_2,\cdots,x_m)^{\mathrm{T}}$，输出结果为 $\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)^{\mathrm{T}}$，，阈值 $\boldsymbol{\theta}=(\theta_1,\theta_2,\cdots,\theta_n)^{\mathrm{T}}$，$w_{ij}$ 为输入 i 同神经元 j 的权数，则第 j 个神经元所接收到的值为

$$z_j=\sum_{i=1}^{m}w_{ij}x_i,y_j=f(z_j-\theta_j),\qquad j=1,2,\cdots,n.\quad(5.2)$$

用矩阵表示为

$$\boldsymbol{W}=\begin{bmatrix}w_{11} & w_{12} & \cdots & w_{1n}\\ w_{21} & w_{22} & \cdots & w_{2n}\\ \cdots & \cdots & \cdots & \cdots\\ w_{m1} & w_{m2} & \cdots & w_{mn}\end{bmatrix},$$

$$Y = f(W^{T}X - \theta). \tag{5.3}$$

5.2 线性网络

最为简单的一类神经网络是线性网络，它是阈网络的一种. 类似(5.1)，可以讨论多种激活函数. 一类最为简单的激活函数是线性函数. 如果一个神经网络的激活函数满足

$$\left.\begin{aligned} f(cz) &= cf(z), \\ f(x+y) &= f(x)+f(y), \end{aligned}\right\} \tag{5.4}$$

其中，c 是一个实常数，x, y, z 为实变量值，则称该神经网络为简单线性网络.

满足(5.4)的一种简单情况是(5.3)满足

$$Y = W^{T}X. \tag{5.5}$$

当(5.5)满足时，每一个神经元满足

$$y_j = (w_{1j}, w_{2j}, \cdots, w_{mj})\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}. \tag{5.6}$$

从简单线性神经网络(5.6)可以看出，其每个神经元的输出只与其相关的权数和输入有关，因此，n 个神经元是相互独立的，所以可以采用单个神经元研究其神经网络的特性.

例 5.1 (5.5)决定的单层线性网络无法识别“异或问题”(XOR——exclusive-or). 异或问题是逻辑中最为简单的一种运算关系，两个逻辑变量 x_1, x_2 的运算及结果 y 满足

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

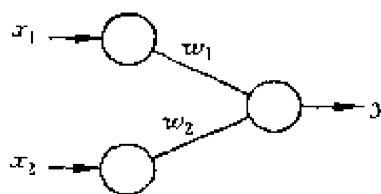


图 5.5 异或问题网络

图 5.5 是求解两变量一个输出的单层线性神经网络. 如果这个神经网络可以求解异或问题,则满足 $y=w_1x_1+w_2x_2$. 求解这个方程得到

$$\begin{cases} \quad\quad w_2 = 1, \\ w_1 \quad\quad = 1, \\ w_1 + w_2 = 0. \end{cases}$$

这是一个矛盾方程,故无法确定权数,使得单层线性神经网络识别异或问题. □

虽说单层线性神经网络有例 5.1 的致命缺陷,但多数研究者仍然继承其研究方法. 下面介绍确定权数的几个常用规则. 这些规则在以后的相关章节中有进一步的推广.

1. Hebb 规则

Hebb 规则是在有监督学习的情况下使用. 设 $\boldsymbol{X}(t)=(x_1(t), x_2(t),\cdots,x_m(t))^{\mathrm{T}}$ 和 $\boldsymbol{Y}(t)=(y_1(t),y_2(t),\cdots,y_n(t))^{\mathrm{T}}$ 是第 t 组用于学习的输入输出数据,w_{ij}的变化为

$$\delta w_{ij}(t) = \varepsilon_t x_i(t) y_j(t), \tag{5.7}$$

其中,ε_t 称为学习效率. (5.7)的含义是通过输入和输出的相关性来体现权数的变化. 当输入和输出的相关性越大,它们的相关权与其成正比. 权的修改公式为

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \delta\boldsymbol{W}(t),\ \delta\boldsymbol{W}(t) = (\delta w_{ij}(t)). \tag{5.8}$$

定理 5.1 当 $\boldsymbol{X}(1),\boldsymbol{X}(2),\cdots,\boldsymbol{X}(m)$为标准正交向量时,若 $\boldsymbol{W}(0)=0$,通过 Hebb 学习规则(5.8)学习 m 次,则由(5.5)决定的单

层线性神经网络可以区分这 m 个向量，且对输入 $\boldsymbol{X}(t)(1\leqslant t\leqslant m)$，认知为 $\varepsilon_t\boldsymbol{Y}(t)$.

证明 在学习 m 次以后，有 $\boldsymbol{Y}(t)=\boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{X}(t)$，满足(5.7)和(5.8)，由(5.7)得到

$$\delta\boldsymbol{W}(t)=(\varepsilon_t x_1(t),\varepsilon_t x_2(t),\cdots,\varepsilon_t x_m(t))^{\mathrm{T}}(y_1(t),y_2(t),\cdots,y_n(t)),$$
$$t=1,2,\cdots,m,$$

再由(5.8)和 $\boldsymbol{W}(0)=\boldsymbol{0}$，得

$$\boldsymbol{W}(m+1)=\boldsymbol{W}(m)+\delta\boldsymbol{W}(m)=\sum_{t=1}^{m}\delta\boldsymbol{W}(t).$$

重新认知 $\boldsymbol{X}(t)$，有

$$\begin{aligned}\boldsymbol{Y}&=\boldsymbol{W}^{\mathrm{T}}(m+1)\boldsymbol{X}(t)=\Big(\sum_{\tau=0}^{m}\delta\boldsymbol{W}(\tau)\Big)^{\mathrm{T}}\boldsymbol{X}(t)\\&=\sum_{\tau=1}^{m}(y_1(\tau),y_2(\tau),\cdots,y_n(\tau))^{\mathrm{T}}(\varepsilon_t x_1(\tau),\varepsilon_t x_2(\tau),\cdots,\varepsilon_t x_m(\tau))\\&\quad\cdot(x_1(t),x_2(t),\cdots,x_m(t))^{\mathrm{T}}\\&=\varepsilon_t\boldsymbol{Y}(t).\end{aligned}$$

由此，证明结论. □

定理 5.1 说明 Hebb 规则对正交的向量组有非常好的识别性能. 这是它的长处. 若识别的向量不具有正交性，则 Hebb 规则的认知性就不一定很好. 例 5.1 就说明了问题. (5.7)的一个修正是

$$\delta w_{ij}(t)=\varepsilon_t x_i(t)(t_j-y_j(t)),\qquad(5.7')$$

其中，t_j 是目标输出，$y_j(t)$是第 j 个神经元的实际输出值. (5.7)中的 $y_j(t)$表示第 t 组学习数据的输出，它是一个已知的值. 不同的是(5.7′)中的 $y_j(t)$是在学习了 $t-1$ 组数据后，得到 $\boldsymbol{W}(t)$，而后由 $\boldsymbol{Y}(t)=\boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{X}(t)$确定，$t_j$ 是期望的目标输出. (5.7′)的 $t_j-y_j(t)$ 表示当学习 $t-1$ 步的输出值同目标值有差别时，输入和 $t_j-y_j(t)$ 相关最大的权数变化最大.

2. 最小二乘学习规则

最小二乘学习规则(LMS——least mean square)也称为线性修正规则(ADALINE——adaptive linear element). 记输入 $\boldsymbol{X}$ 和理想输出 $\boldsymbol{D}=(d_1,d_2,\cdots,d_n)^{\mathrm{T}}$,LMS 是在学习的过程中使得

$$F(\boldsymbol{W})=(\boldsymbol{D}-\boldsymbol{Y})^{\mathrm{T}}(\boldsymbol{D}-\boldsymbol{Y}), \tag{5.9}$$

最小,其中 $\boldsymbol{Y}=\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}$.

由前面的讨论,单层线性前向神经网络层的多个神经元的研究可以按一个神经元研究. 求解满足(5.9)的 $\boldsymbol{W}$ 取决所有可能的样本. 通过所有的样本计算求解(5.9)是不经济的,也是不现实的. 对单层线性神经网络可以通过概率分析求解 $\boldsymbol{W}$. 只讨论一个神经元的情形.

设输入 $\boldsymbol{X}=(X_1,X_2,\cdots,X_m)^{\mathrm{T}}$ 和理想输出 $\boldsymbol{d}$ 为随机变量, LMS 的期望模型为

$$F_E(\boldsymbol{W})=E(\boldsymbol{d}-\boldsymbol{Y})^2.$$

变形为

$$\begin{aligned}F_E(\boldsymbol{W})&=E(\boldsymbol{d}-\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X})^2=E(\boldsymbol{d}^2-2\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{d}+\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{W})\\&=E(\boldsymbol{d}^2)-2\boldsymbol{W}^{\mathrm{T}}E(\boldsymbol{d}\boldsymbol{X})+\boldsymbol{W}^{\mathrm{T}}E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})\boldsymbol{W}.\end{aligned}$$

这是一个二次型,最小值存在的充分条件为 $E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})$正定,其中

$$E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})=(E(X_iX_j))_{m\times m}.$$

$E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})$正定的一个充分条件是:$X_i$ 与 $X_j(i\neq j)$相互独立, $E(X_i)=0,i=1,2,\cdots,m$ 和 $E(X_i^2)\neq 0,i=1,2,\cdots,m$.

在 $E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})$正定的条件下,由

$$\frac{\partial F_E}{\partial \boldsymbol{W}}=\begin{bmatrix}\dfrac{\partial F_E}{\partial w_1}\\\dfrac{\partial F_E}{\partial w_2}\\\vdots\\\dfrac{\partial F_E}{\partial w_m}\end{bmatrix}=-2E(\boldsymbol{d}\boldsymbol{X})+2E(\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}})\boldsymbol{W}=0,$$

得

$$W^* = E(XX^{\mathrm{T}})^{-1}E(dX). \tag{5.10}$$

有多个神经元的情况下，理想输出为 $D=(d_1,d_2,\cdots,d_n)^{\mathrm{T}}$. 由(5.10)第 j 个神经元的权为

$$W_j^* = \begin{bmatrix} w_{1j} \\ w_{2j} \\ \vdots \\ w_{mj} \end{bmatrix} = E(XX^{\mathrm{T}})^{-1}E(d_jX), \quad j = 1,2,\cdots,n. \tag{5.10'}$$

上面的讨论说明在一定的条件下(5.9)的最优解是存在的，形式如(5.10)或(5.10′).

在大多数情况下，求解 $E(XX^{\mathrm{T}})$是比较难实现的. 因此需要更一般的数值求解方法. 下面介绍单样本 LMS 学习规则.

从理论分析上得到(5.9)的最优解在期望值模型下是存在的. 一般情况下，不易求解(5.10)的期望值，解决的办法是数值求解. 我们先考虑一次只给出一组输入输出学习数据，称为单个随机样本学习方法. 设 $X=(x_1,x_2,\cdots,x_m)^{\mathrm{T}}$ 和 $D=(d_1,d_2,\cdots,d_n)^{\mathrm{T}}$ 分别是随机选取的一对学习输入输出向量，$Y=W^{\mathrm{T}}X$ 是实际输出向量. 由(5.9)

$$F(W) = \sum_{j=1}^{n}\Big(\sum_{l=1}^{m} w_{lj}x_l - d_j\Big)^2.$$

进而

$$\frac{\partial F(W)}{\partial w_{ij}} = 2x_i\Big(\sum_{l=1}^{m} w_{lj}x_l - d_j\Big) = 2(y_j - d_j)x_i.$$

$F(W)$下降的方向为

$$\delta w_{ij} = -\frac{\varepsilon}{2}\frac{\partial F(W)}{\partial w_{ij}} = \varepsilon(d_j - y_j)x_i,$$

$$i = 1,2,\cdots,m; j = 1,2,\cdots,n, \tag{5.11}$$

其中，ε 为学习效率. (5.11)同 Hebb 规则(5.7′)相同，输入 x_i 和输

出 y_j 相互具有线性修正关系. 因此,LMS、Hebb 规则、Adaline 规则有时混称. 它们可以统称为误差修正规则. 于是权数的变化形式为

$$\begin{aligned} W(t+1) &= W(t) + \delta W(t) \\ &= W(t) + \varepsilon_t \{ (d_j(t) - y_j(t)) x_i(t) \}_{m\times n}. \end{aligned} \quad (5.12)$$

从推导中得知(5.11)是函数 $F(W)$ 下降的方向. 我们最终的目的是得到满足(5.10′)的 W^*. 通过一个个单一随机样本的学习, 在什么样的条件下(5.12)可以收敛到 W^*? Widrow 和 Stearns 在[7]中给出收敛的充分条件为

$$\sum_{t=1}^{\infty} \varepsilon_t = \infty, \qquad \sum_{t=1}^{\infty} \varepsilon_t^2 < \infty. \quad (5.13)$$

很明显, $\varepsilon_t = \frac{1}{t}$ 满足(5.13).

上面的情形可以推广到每次采用随机选取一组样本进行学习的情况. 设一组样本为

$$\{(X^1, D^1), (X^2, D^2), \cdots, (X^K, D^K)\},$$

其中, $X^k = (x_1^k, x_2^k, \cdots, x_m^k)^T$ 和 $D^k = (d_1^k, d_2^k, \cdots, d_n^k)^T$. 目标函数为

$$F(W) = \sum_{k=1}^{K} (Y^k - D^k)^T (Y^k - D^k),$$

其中, $Y^k = W^T X^k$. 权数的计算公式为

$$\begin{aligned} W(t+1) &= W(t) + \delta W(t) \\ &= W(t) + \varepsilon_t \sum_{k=1}^{K} \{ (d_j^k(t) - y_j^k(t)) x_i^k(t) \}_{m\times n}. \end{aligned}$$

一组样本的学习方法介于单个样本和大规模样本的学习方法之间. 于是 K 的选取决定于收敛的特性.

读者应该注意评价 Hebb 规则和 LMS 期望模型中解的区别. Hebb 规则的评价出发点是考虑每一个体,因此,有例 5.1 无法识别异或问题的最坏情况和定理 5.1 的可以区别 m 个标准正交向量的理想结论. 回想第一章对算法的评价方法,三种评价方法中有

两种为：最坏分析法和概率分析法. 对 Hebb 规则的评价就是最坏分析法. LMS 期望模型解的评价方法就是概率分析法. 在概率分析有最优解的情况下，还需用确定性的计算方法近似求解.

5.2.2 非线性网络

当激活函数选非线性函数时，神经网络称为非线性神经网络. 非线性神经网络也是阈网络的一种. 常见的非线性函数有

符号函数(见图 5.6(a))：$f(z)=\begin{cases}1, & z>T,\\ 0, & \text{其他},\end{cases}$

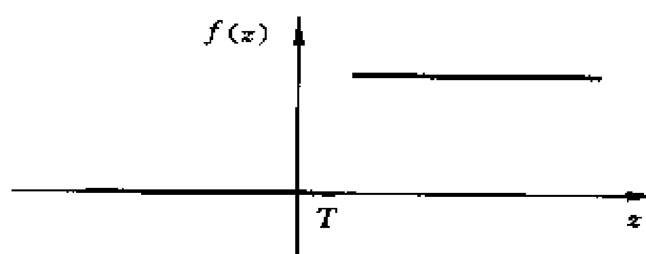


图 5.6(a) 符号函数

S(Sigmold)形函数(见图(5.6(b))：$f(z)=\dfrac{1}{1+e^{-z}}$

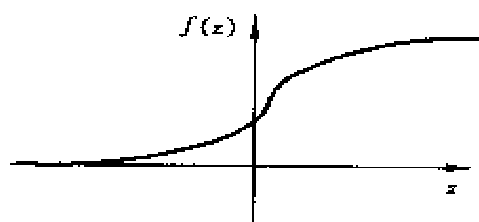


图 5.6(b) S 形函数

符号函数是阶跃函数，在物理中容易实现. 在识别和归类问题中，当输出值为 1 时，我们可以肯定地说出输入的归类. 它的缺点是数学性质较差，如在 T 点不光滑. S 形函数弥补了这一方面的不

足，使得函数值在(0,1)区间连续变化. S 形函数又称 Sigmoid 函数. 从符号函数和 S 形函数及神经网络结构可以看出，前向单层非线性神经网络非常适合分类和判定问题，如以输出值 0 和 1 各表示分类问题或以输出为"否"和"是"判定问题. 由于符号函数和 S 形函数本身所具有的特性，在实际应用中多采用符号函数，因为这个函数的值有明确的 0 和 1 归类；理论分析中常采用后一个函数，因为这个函数的数学性质非常好.

先研究符号函数的性质. 从下例可以看到，符号函数决定的单层非线性神经网络同样无法实现 XOR 功能.

例 5.2　采用符号函数的单层非线性神经网络同样无法实现 XOR 功能. 从例 5.1 得到

x_1	x_2	z	期望输出
0	0	0	0
0	1	w_2	1
1	0	w_1	1
1	1	w_1+w_2	0

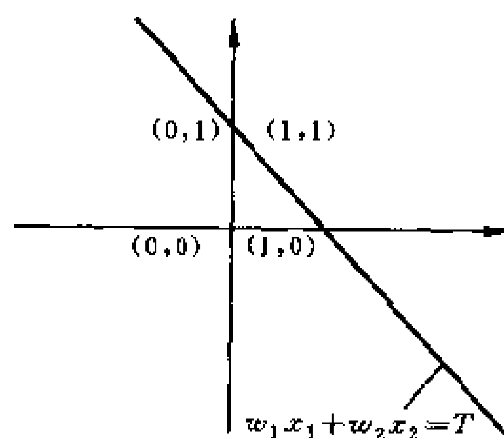


图 5.7　$w_1x_1+w_2x_2=T$ 分割平面图

由 $w_1x_1+w_2x_2=T$，其中 T 为阈值，将平面分成两部分(见图

5.7). 如果具有区分功能，则满足(0,0)和(1,1)在一半平面内的同时，(0,1)和(1,0)在另一半平面内，即要求

$$(0,0),(1,1) \in H1 = \{(x_1,x_2) \mid w_1x_1 + w_2x_2 \leqslant T\},$$

且

$$(0,1),(1,0) \in H2 = \{(x_1,x_2) \mid w_1x_1 + w_2x_2 > T\},$$

或反之. 这样要求

$$\begin{cases} w_1 + w_2 \leqslant T, \\ \qquad w_2 > T, \\ \qquad w_1 > T, \end{cases}$$

的 w_1, w_2, T 是不存在的，因此，结论成立. □

以单个神经元单一随机样本讨论学习规则. 符号函数决定的神经网络的学习规则同线性网络的学习规则相同. 设给定初始 $\boldsymbol{W}(0)$ 和阈值 T_0，学习规则是

$$\boldsymbol{W}(t+1) = \varepsilon_t\{d(t) - y(t)\}\boldsymbol{X}(t) + \boldsymbol{W}(t). \tag{5.14}$$

从(5.14)可以得到

(1) 当 $d(t)=y(t)$ 时，表明已经识别第 t 组输入 $\boldsymbol{X}(t)$，此时，$\boldsymbol{W}(t+1)=\boldsymbol{W}(t)$，即权数不修改. 此时阈值也不修改；

(2) 当 $d(t)=1$ 且 $y(t)=0$ 时，神经网络没有区分输入 $\boldsymbol{X}(t)$，此时表明

$$z(t) = \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{X}(t) \leqslant T_t,$$

于是(5.14)修改权数使得

$$\begin{aligned} z(t+1) &= \boldsymbol{W}^{\mathrm{T}}(t+1)\boldsymbol{X}(t) \\ &= \varepsilon_t\boldsymbol{X}^{\mathrm{T}}(t)\boldsymbol{X}(t) + \boldsymbol{W}^{\mathrm{T}}(t)\boldsymbol{X}(t) > z(t), \end{aligned}$$

将阈值修改为

$$T_{t+1} = T_t - \varepsilon_t T_t; \tag{5.15}$$

(3) 当 $d(t)=0$ 且 $y(t)=1$ 时，神经网络没有区分输入 $\boldsymbol{X}(t)$，此时表明

$$z(t) = W^{T}(t)X(t) > T_t,$$

于是(5.14)修改权数使得

$$z(t+1) = W^{T}(t+1)X(t) = -\varepsilon_t X^{T}(t)X(t) + W^{T}(t)X(t) < z(t),$$

将阈值修改为

$$T_{t+1} = T_t + \varepsilon_t T_t. \tag{5.16}$$

从理论来考虑上述学习规则的收敛性. 符号函数 $f(u(t))$ 中的变元为

$$u(t) = W^{T}(t)X(t) - T_t. \tag{5.17}$$

在学习过程中,$W(t)$和阈值 T_t 都是待确定的参数. 若记广义权和输入为

$$\overline{W(t)} = (W(t), T_t)^{T}, \quad \overline{X(t)} = (X(t), -1)^{T},$$

(5.17)变形为

$$u(t) = (\overline{W(t)})^{T}\,\overline{X(t)}. \tag{5.18}$$

这是一个 $n+1$ 个神经元的单层神经网络问题,其阈值为 0.

如果给定的分类问题或判定问题是线性可分的,即如图 5.8 所示,存在一条直线(平面)将平面(空间)分为两部分,使得理想输出为 1 和 0 的输入点分别在各自一半的部分中.

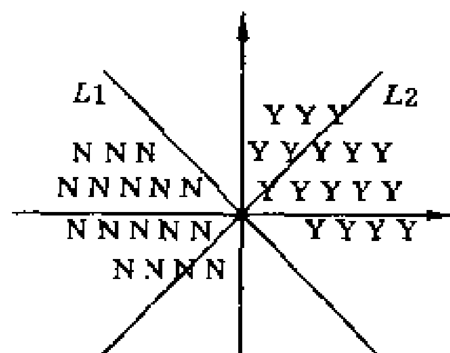


图 5.8　线性可分示意图

(5.16)过原点,由上面(1)、(2)和(3)的讨论,当已有的权数使得神经网络不能区分"Y"和"N"两部分时,即如图 5.8 中权数确定

的直线(平面)$L2:u(t)=W^{T}(t)X-T_t$ 时,从理论上来说,只要按一定的条件,用(5.14)和(5.15)或(5.16)修改广义权和适当选取 ϵ_t, $L2$ 最终逼近 $L1$.一个直观修改广义权的办法是:先以理想输出为1的输入数据学习,使得神经网络可以识别理想输出为1的数组.在学习的过程中要求适当选取 ϵ_t 使得直线 $L1$ 不能变化太快,以避免将已经识别的数据重新不能识别,且至少使一个没有识别的数组得以识别.在理想输出为1的输入数据得以识别后,再用类似的方法,识别理想输出为0的输入数据.由此可以有限步达到识别线性可分问题.总结上面的讨论,得到这样的结论:如果给定的分类问题或判定问题是线性可分的且 ϵ_t 选取合适,则(5.14)的学习规则使得神经网络可以识别这些问题.

上面讨论了一层线性与非线性神经网络,下面将介绍一个多层神经网络的模型、学习规则和算法.

5.3 多层前向神经网络

以一个两层的神经网络图5.9来理解多层前向神经网络(multi-layer feed forward neural networks).

同单层神经网络相同的解释:输入层中的元素为轴突;输出层的元素为神经元的内核;隐层的元素也为神经元的内核.由此得到一个两层的神经元.它的输入层有3个元素,隐层有两个神经元,输出层有4个神经元.为了方便起见,各层的元素都称为神经元.前向神经网络要求每一层内的神经元无相连关系且上一层(左边)的输出是下一层(右边)的输入.

一般的多层前向神经网络的结构如图5.10所示.它们有两层以上(包括两层)的神经元,第 i 层的神经元的输出为第 $i+1$ 层神经元的输入,但在一层之内的神经元是没有输入输出关系.除输入和输出层以外的神经元层称为隐层.输入层记为第0层,输出层所

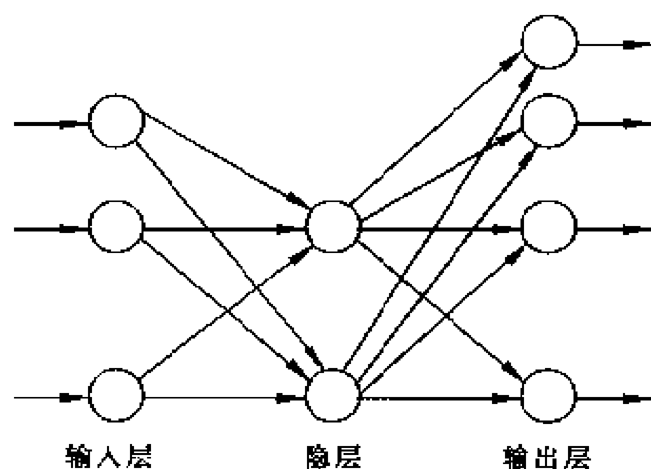


图 5.9　三层前向神经网络结构示意图

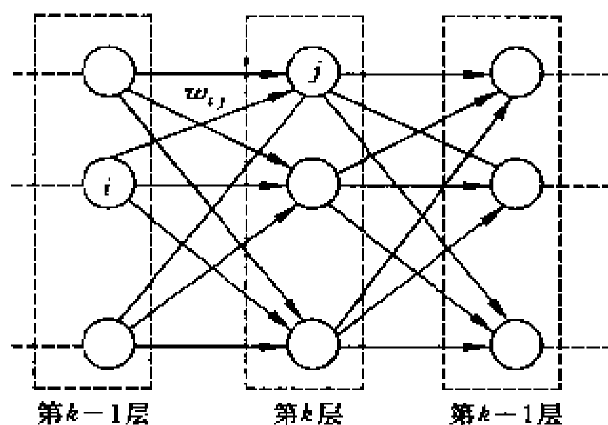


图 5.10　多层前向神经网络结构示意图

在的层数为神经网络的层数.

记第 $k-1$ 层到第 k 层的权数矩阵为

$$\boldsymbol{W}_k = (w_{ij}^k)_{n_{k-1}\times n_k},$$

其中，n_k 表示第 k 层神经元个数. 输入向量为 $\boldsymbol{X}=(x_1,x_2,\cdots,x_{n_0})^{\mathrm{T}}$.

若多层神经网络的每一层都采用(5.5)的激活函数输出，称为多层简单线性前向网络，则第 n 层的输出为

$$Z_n = (W_n)^T(W_{n-1})^T\cdots(W_1)^T X.$$

因此,多层简单线性前向网络同(5.5)等价,无法区别 XOR 问题.

5.3.1 非线性神经网络

采用非线性激活函数的神经网络,它区别异或问题的效果又如何?

例 5.3 设两层前向神经元网络如图 5.11,

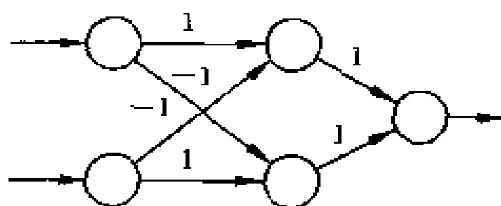


图 5.11 区别 XOR 问题的神经网络

激活函数采用符号函数,阈值为 0. 我们讨论图 5.11 的神经网络是否能够区别 XOR 问题.

由

$$W_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad W_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

当 $X=\begin{bmatrix} 1 \\ 0 \end{bmatrix}$时,

$$Z_1 = W_1^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, Y_1 = f(Z_1) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

$$Z_2 = W_2^T Y_1 = 1, \quad Y_2 = f(Z_2) = 1.$$

当 $X=\begin{bmatrix} 0 \\ 1 \end{bmatrix}$时,

$$Z_1 = W_1^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, Y_1 = f(Z_1) = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$Z_2 = W_2^T Y_1 = 1, \quad Y_2 = f(Z_2) = 1.$$

当 $X=\begin{bmatrix}1\\1\end{bmatrix}$时，

$$Z_1=W_1^T\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}, Y_1=f(Z_1)=\begin{bmatrix}0\\0\end{bmatrix},$$

$$Z_2=W_2^TY_1=0,\quad Y_2=f(Z_2)=0.$$

当 $X=\begin{bmatrix}0\\0\end{bmatrix}$时，

$$Z_1=W_1^T\begin{bmatrix}0\\0\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}, Y_1=f(Z_1)=\begin{bmatrix}0\\0\end{bmatrix},$$

$$Z_2=W_2^TY_1=0,\quad Y_2=f(Z_2)=0.$$

可以看出，由阈值为 0 的符号函数决定的图 5.11 神经网络能够区别 XOR 问题. □

虽说采用符号函数的多层前向神经网络有很强的分类功能，如对 XOR 问题的识别，但因没有很好的学习方法，故在实际问题中应用较少. 而采用线性函数的多层前向神经网络，由于同单层线性神经网络的功效等价，因此，无法解决复杂问题的分类或识别问题.

S 形函数有很好的函数特性，其效果又近似符号函数，因此现主要讨论采用 S 形函数的前向多层神经网络的学习方法.

假设一个 K 层神经网络每一层的输入输出关系如下：从第 0 层到第一层的原始输入向量、权矩阵、接受值向量和输出向量及它们之间的关系分别为

$$X=\begin{bmatrix}x_1\\x_2\\\vdots\\x_{n_0}\end{bmatrix}, W_1=(w_{ij}^1)_{n_0\times n_1}, Z_1=\begin{bmatrix}z_1^1\\z_2^1\\\vdots\\z_{n_1}^1\end{bmatrix}=W_1^TX, Y_1=\begin{bmatrix}y_1^1\\y_2^1\\\vdots\\y_{n_1}^1\end{bmatrix}=f(Z_1),\tag{5.19}$$

第 $k-1$ 层到第 k 层的权矩阵、接受值向量和输出向量及它们

之间的关系分别为

$$\boldsymbol{W}_k=(w_{ij}^k)_{n_{k-1}\times n_k},\boldsymbol{Z}_k=\begin{bmatrix}z_1^k\\z_2^k\\\vdots\\z_{n_k}^k\end{bmatrix}=\boldsymbol{W}_k^{\mathrm{T}}\boldsymbol{Y}_{k-1},\boldsymbol{Y}_k=\begin{bmatrix}y_1^k\\y_2^k\\\vdots\\y_{n_k}^k\end{bmatrix}=f(\boldsymbol{Z}_k),\tag{5.20}$$

其中，$y_i^k=f(z_i^k)$.

类似 5.2 节对单层线性网络学习规则的讨论方法，我们先讨论单样本学习规则. 学习的原则是：确定 $\boldsymbol{W}$，使得

$$F(\boldsymbol{W})=(\boldsymbol{D}-\boldsymbol{Y}_K)^{\mathrm{T}}(\boldsymbol{D}-\boldsymbol{Y}_K)\tag{5.21}$$

最小，其中，$\boldsymbol{D}=(d_1,d_2,\cdots,d_{n_K})^{\mathrm{T}}$ 为理想输出. 分输出层、隐层及输入层对学习规则讨论.

一、输出层

对输出层的每一个神经元，由(5.20)得

$$\left.\begin{aligned}&\frac{\partial F}{\partial w_{ij}^K}=-2(d_j-y_j^K)\frac{\partial y_j^K}{\partial w_{ij}^K},\\&\frac{\partial y_j^K}{\partial w_{ij}^K}=\frac{dy_j^K}{dz_j^K}y_i^{K-1},\quad i=1,2,\cdots,n_{K-1},\quad j=1,2,\cdots,n_K.\end{aligned}\right\}\tag{5.22}$$

二、隐层及输入层

对第 k 层($k\leqslant K-1$)的一个神经元 j，当 $k=K-1$ 时，

$$\begin{aligned}\frac{\partial F}{\partial w_{ij}^{K-1}}&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{\partial y_{l_K}^K}{\partial w_{ij}^{K-1}}\\&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\frac{\partial z_{l_K}^K}{\partial w_{ij}^{K-1}}\\&=-2\sum_{l_K=1}^{n_K}(d_{l_K}-y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^K\frac{\partial y_{l_{K-1}}^{K-1}}{\partial w_{ij}^{K-1}}\end{aligned}\tag{5.23}$$

$$= -2\sum_{l_K=1}^{n_K}(d_{l_K} - y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}w_{jl_K}^K\frac{dy_j^{K-1}}{dz_j^{K-1}}y_i^{K-2},$$

$$i = 1,2,\cdots,n_{K-2};\quad j = 1,2,\cdots,n_{K-1}.$$

图 5.12 给出(5.23)各参数的关系.

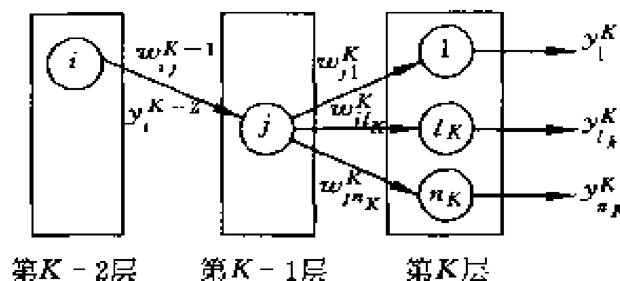


图 5.12 (5.23)的参数关系

对其他的隐层同(5.23)式有相同的结论. 当 $k\leqslant K-2$ 时,有

$$\frac{\partial F}{\partial w_{ij}^k} = -2\sum_{l_K=1}^{n_K}(d_{l_K} - y_{l_K}^K)\frac{dy_{l_K}^K}{dz_{l_K}^K}\sum_{l_{K-1}=1}^{n_{K-1}}w_{l_{K-1}l_K}^K\frac{dy_{l_{K-1}}^{K-1}}{dz_{l_{K-1}}^{K-1}}\cdots$$

$$\sum_{l_{k+2}=1}^{n_{k+2}}w_{l_{k+2}l_{k+3}}^{k+3}\frac{dy_{l_{k+2}}^{k+2}}{dz_{l_{k+2}}^{k+2}}\sum_{l_{k+1}=1}^{n_{k+1}}w_{l_{k+1}l_{k+2}}^{k+2}\frac{dy_{l_{k+1}}^{k+1}}{dz_{l_{k+1}}^{k+1}}w_{jl_{k+1}}^{k+1}\frac{dy_j^k}{dz_j^k}y_i^{k-1},\quad(5.24)$$

$$i = 1,2,\cdots,n_{k-1};\quad j = 1,2,\cdots,n_k.$$

其中,输入层 $k=0$ 时 $y_i^0=x_i, 1\leqslant i\leqslant n_0$.

由于(5.22)、(5.23)和(5.24)比较难以记忆,可以用矩阵的形式表示. 记

$$\boldsymbol{B}_{K+1} = \begin{bmatrix} d_1 - y_1^K \\ d_2 - y_2^K \\ \vdots \\ d_{n_K} - y_{n_K}^K \end{bmatrix}, \boldsymbol{W}_{K+1} = \boldsymbol{I}(\text{单位阵}),$$

则(5.21)为

$$\boldsymbol{F}(\boldsymbol{W}) = (\boldsymbol{B}_{K+1})^{\mathrm{T}}\boldsymbol{B}_{K+1}.\quad(5.21')$$

记

$$\boldsymbol{B}_K = \operatorname{diag}\left\{\frac{dy_1^K}{dz_1^K}, \frac{dy_2^K}{dz_2^K}, \cdots, \frac{dy_{n_K}^K}{dz_{n_K}^K}\right\} \boldsymbol{W}_{K+1} \boldsymbol{B}_{K+1},$$

其中，$\operatorname{diag}(a_1, a_2, \cdots, a_n)$表示以 $a_1, a_2, \cdots, a_n$ 为对角元素的 n 阶对角阵. (5.22)改写为

$$\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^K}\right)_{n_{K-1}\times n_K} = -2\begin{bmatrix} y_1^{K-1} \\ y_2^{K-1} \\ \vdots \\ y_{n_{K-1}}^{K-1} \end{bmatrix} (\boldsymbol{B}_K)^{\mathrm{T}}. \qquad (5.22')$$

记

$$\boldsymbol{B}_{K-1} = \operatorname{diag}\left\{\frac{dy_1^{K-1}}{dz_1^{K-1}}, \frac{dy_2^{K-1}}{dz_2^{K-1}}, \cdots, \frac{dy_{n_{K-1}}^{K-1}}{dz_{n_{K-1}}^{K-1}}\right\} \boldsymbol{W}_K \boldsymbol{B}_K,$$

则(5.23)改写成

$$\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^{K-1}}\right)_{n_{K-2}\times n_{K-1}} = -2\begin{bmatrix} y_1^{K-2} \\ y_2^{K-2} \\ \vdots \\ y_{n_{K-1}}^{K-2} \end{bmatrix} (\boldsymbol{B}_{K-2})^{\mathrm{T}}. \qquad (5.23')$$

记

$$\boldsymbol{B}_K = \operatorname{diag}\left\{\frac{dy_1^k}{dz_1^k}, \frac{dy_2^k}{dz_2^k}, \cdots, \frac{dy_{n_k}^k}{dz_{n_k}^k}\right\} \boldsymbol{W}_{k+1} \boldsymbol{B}_{k+1},$$

则(5.24)改写为

$$\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^k}\right)_{n_{k-1}\times n_k} = -2\begin{bmatrix} y_1^{k-1} \\ y_2^{k-1} \\ \vdots \\ y_{n_{k-1}}^{k-1} \end{bmatrix} (\boldsymbol{B}_k)^{\mathrm{T}}. \qquad (5.24')$$

当第 t 个学习样本值 $X(t)$输入后，可以依次得到

$$\boldsymbol{Y}_1(t) = (y_1^1(t), y_2^1(t), \cdots, y_{n_1}^1(t))^{\mathrm{T}},$$

$$Y_2(t) = (y_1^2(t), y_2^2(t), \cdots, y_{n_2}^2(t))^{\mathrm{T}},$$

$$\cdots$$

$$Y_K(t) = (y_1^K(t), y_2^K(t), \cdots, y_{n_K}^K(t))^{\mathrm{T}},$$

此时，用(5.22′)(5.23′)(5.24′)和已知的 $\boldsymbol{W}(t)$ 可以得到相应的值，用带有参数 t 的形式表示为 $\boldsymbol{B}_k(t)$ 和 $\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}$，按 $F(\boldsymbol{W})$ 下降的方向确定权数的学习规则为

$$\boldsymbol{W}_k(t+1) = \boldsymbol{W}_k(t) + \delta\boldsymbol{W}_k(t), \quad k = K, K-1, \cdots, 1, \tag{5.25}$$

其中，

$$\delta\boldsymbol{W}_k(t) = -\frac{1}{2}\varepsilon_t\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k} = \varepsilon_t\begin{bmatrix} y_1^{k-1}(t) \\ y_2^{k-1}(t) \\ \vdots \\ y_{n_{K-1}}^{k-1}(t) \end{bmatrix}(\boldsymbol{B}_k(t))^{\mathrm{T}},$$

ε_t 为第 t 步的学习效率.

从(5.25)的学习规则看到这样一个规律：$\boldsymbol{W}_k$ 的修正是从最后一层神经元的权数 $\boldsymbol{W}_K$ 开始，反向递推求解修正 $K-1$ 层的 $\boldsymbol{W}_{K-1}$，…，一直修正到第一层的权数 $\boldsymbol{W}_1$. 由此将(5.25)学习规则称为反推(BP)学习规则(backpropagation). 采用 S 形函数的前向多层神经网络有下面的反推学习算法，这是一个有监督的学习算法.

反推学习算法

STEP1　选定学习的数组 $\{\boldsymbol{X}(t), \boldsymbol{D}(t)\}$，$t=1,2,\cdots,T$，随机确定初始权矩阵 $\boldsymbol{W}(0)$；

STEP2　用学习数据 $\boldsymbol{X}(t)$ 计算 $\boldsymbol{Y}_1(t), \boldsymbol{Y}_2(t), \cdots, \boldsymbol{Y}_K(t)$；

STEP3　用(5.22′)(5.23′)(5.24′)的计算，反向修正 $\boldsymbol{W}(t)$，修正公式为(5.25)；直到学习完所有的数组.

当激活函数 $f(x)=\dfrac{1}{1+e^{-x}}$时，

$$\frac{\mathrm{d}y_j^k}{\mathrm{d}z_j^k}=f(z_i^k)(1-f(z_j^k)),\quad i=1,2,\cdots,n_{k-1},\quad j=1,2,\cdots,n_k,\tag{5.26}$$

代入(5.22′)(5.23′)(5.24′)使计算得以简化.

BP 算法弥补了符号函数在实际应用中难以确定权数的不足，使得具有很强识别功能的前向多层神经网络得以应用. 但应用 BP 算法中需要注意以下主要问题：

(1) 激活函数为 S 形函数，只能趋近 0 或 1. 于是，在神经网络的工作期，输出值只能接近 1 或 0. 判定一个数组的归类问题就需要给出一个分界值，如 0.5，而不是绝对的 1 或 0.

(2) $\boldsymbol{W}(0)$的选取最好是随机的. 当 $\boldsymbol{W}(0)$的所有权数相等时，由(5.20)的 $\boldsymbol{Z}_k=(\boldsymbol{W}_k)^{\mathrm{T}}\boldsymbol{Y}_{k-1},k=1,2,\cdots,K$ 推出：无论什么样的输入 $\boldsymbol{X}$，$\boldsymbol{Z}_1$ 中的全部分量相同；当第一层的激活函数取同一个函数时，$\boldsymbol{Y}_1$ 的全部分量相同；当每一层的激活函数取同一个函数时，由于 $\boldsymbol{Z}_k$ 的各分量相同，所以，$\dfrac{\mathrm{d}y_1^k}{\mathrm{d}z_1^k}=\cdots=\dfrac{\mathrm{d}y_{n_k}^k}{\mathrm{d}z_{n_k}^k}$，再由(5.22′)(5.23′)(5.24′)的运算得$\left(\dfrac{\partial F(\boldsymbol{W})}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}$中的所有元素相等. 由此得到的修正权对很多问题无法识别.

(3) 算法的全局最优性. 这里存在的理论问题之一是目标函数的选取及应满足的条件，如采用期望目标模型在什么条件下有唯一的极值点？之二是一般的情况下是否有唯一的极值点？

只要学习效率 ε_t 选取合适，(5.22′)、(5.23′)和(5.24′)决定的下降算法，类似(5.13)，收敛到局部最优，可参考文献[8]. 为了避免局部最优解的一个改进方法是将(5.25)的学习规则修正为

$$\delta\boldsymbol{W}_k(t)=-\frac{1}{2}\varepsilon_t\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^k}(t)\right)_{n_{k-1}\times n_k}+\alpha_t\delta\boldsymbol{W}_k(t-1),\tag{5.27}$$

(5.27)右端的第二项相当于一个势能"惯性",式中的 α_t 取决于权数两次变化的相互关系.

(4) 对于多样本学习,即一次是用一组样本学习,此时的学习规则是使

$$F(W) = \sum_{t=1}^{P}(D(t) - Y_K(t))^{\mathrm{T}}(D(t) - Y_K(t))$$

最小. 由于 $F(W) = \sum_{t=1}^{P}F_t(W)$, 其中

$$F_t(W) = (D(t) - Y_K(t))^{\mathrm{T}}(D(t) - Y_K(t)),$$

所以学习规则是将 P 个样本用(5.25)的方法累加.

5.3.2 BP 算法的一个示例

为了说明 BP 算法的计算过程,用图 5.13 的共有 6 个神经元的两层前向神经网络[9]去产生图 5.14 的理想输出.

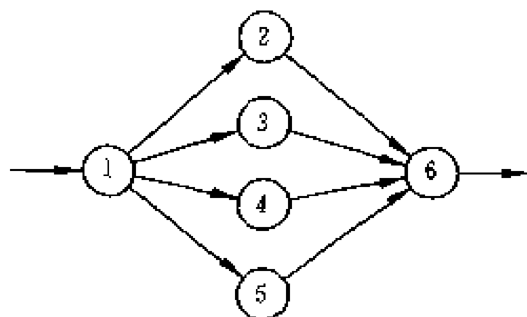


图 5.13 6 个神经元的两层神经网络

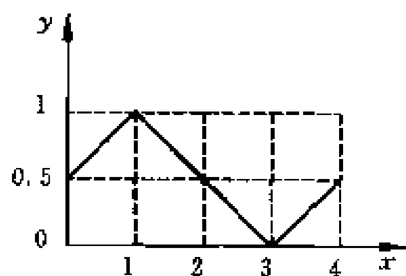


图 5.14 理想输出

权矩阵随机选取为

$$w_{12} = 0.2,\ w_{13} = 0.3,\ w_{14} = 0.4,\ w_{15} = 0.5,$$

$$w_{26} = 0.5,\ w_{36} = 0.2,\ w_{46} = 0.1,\ w_{56} = 0.4.$$

学习数据如表 5.1,按输入变量每 0.05 为一个分割点,共有 80 个数组.

表 5.1　学 习 数 据

输入值	输出值	输入值	输出值	输入值	输出值
0.000	0.500	1.800	0.600	2.900	0.050
0.050	0.525	1.850	0.575	2.950	0.025
0.100	0.550	1.900	0.550	3.000	0.000
0.150	0.575	1.950	0.525	3.050	0.025
0.200	0.600	2.000	0.500	3.100	0.050
…	…	2.050	0.475	…	…
0.900	0.950	2.100	0.450	3.800	0.400
0.950	0.975	2.150	0.425	3.850	0.425
1.000	1.000	2.200	0.400	3.900	0.450
1.050	0.975	2.250	0.375	3.950	0.475
1.100	0.950	…	…	4.000	0.500
…	…	…	…		

激活函数采用

$$f(x)=\frac{1}{1+e^{-x}}.$$

(5.25)的学习效率 $\epsilon_t=1$. 从表 5.1 的开始对每一个数据进行学习.

第一次学习时:

$$x_1=0,$$

$$z_2=z_3=z_4=z_5=0,$$

$$y_i=f(z_i)=\frac{1}{2},\quad i=2,3,4,5,$$

$$z_6=\frac{1}{2}(0.5+0.2+0.1+0.4)=0.6,$$

$$y_6=0.6457.$$

反推确定第二层权数变化:

$$\boldsymbol{B}_3=(0.5-0.6457)=(-0.1457),$$

$$\boldsymbol{B}_2=\frac{\mathrm{d}y_6}{\mathrm{d}z_6}B_3=0.6457\times(1-0.6457)\times(-0.1457)$$
$$=-0.0333.$$

梯度矩阵

$$\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^2}\right)_{6\times 1}=-2\begin{bmatrix}y_2\\y_3\\y_4\\y_5\end{bmatrix}(\boldsymbol{B}_2)^{\mathrm{T}}=-2\begin{bmatrix}-0.0167\\-0.0167\\-0.0167\\-0.0167\end{bmatrix},$$

由(5.25)的学习规则得

$$\begin{bmatrix}w_{26}\\w_{36}\\w_{46}\\w_{56}\end{bmatrix}:=\begin{bmatrix}0.5\\0.2\\0.1\\0.4\end{bmatrix}-\frac{1}{2}\left(\frac{\partial F(\boldsymbol{W})}{\partial w_{ij}^2}\right)_{6\times 1}=\begin{bmatrix}0.4833\\0.1833\\0.0833\\0.3833\end{bmatrix}.$$

反推确定第一层的权数变化：

$$\boldsymbol{B}_1=\mathrm{diag}(0.5\times(1-0.5),0.25,0.25,0.25)\begin{bmatrix}w_{26}\\w_{36}\\w_{46}\\w_{56}\end{bmatrix}(\boldsymbol{B}_2)^{\mathrm{T}}$$

$$=\mathrm{diag}(0.25,0.25,0.25,0.25)\begin{bmatrix}0.5\\0.2\\0.1\\0.4\end{bmatrix}(-0.0333)^{\mathrm{T}}$$

$$=\begin{bmatrix}-0.0042\\-0.0017\\-0.0001\\-0.0003\end{bmatrix}.$$

$$\left(\frac{\partial F(W)}{\partial w_{ij}^1}\right)_{1\times 4} = -2x(B_1)^T = -2x\begin{bmatrix}-0.0167\\-0.0167\\-0.0167\\-0.0167\end{bmatrix} = 0,$$

再由(5.25)的学习规则得

$$(w_{12}, w_{13}, w_{14}, w_{15})^T = (0.2, 0.3, 0.4, 0.5)^T.$$

通过这一个样本的学习，再输入这个样本，则有

$$z_6 = 0.5(0.4833 + 0.1833 + 0.0833 + 0.3833)$$
$$= 0.5666, y_6 = 0.6380,$$

比原有的 0.6457 下降 5.3%. 以表 5.1 的数据，按顺序依次进行学习，图 5.15 和图 5.16 分别表示在学习总数为 1493 和30 588时的示意图形.



图 5.15　学习 1 493 次后示意图



图 5.16　学习 30 588 次后的示意图

前向多层神经网络的实际应用包括很多方面，如手写体的识别，语音识别，文本语音转换，图象识别，生物和医学信号处理等[10].

5.4　竞争学习神经网络

竞争学习神经网络是无监督神经网络中的一类. 给定输入数据，不同于有监督学习中需要对应的理想输出，无监督学习不需要对应的理想输出. 也就是说，它的输出结果表明输入数据中的相关

或相似性.

对于单层前向网络,一类简单的线性竞争学习模型为

$$y_j = \begin{cases} 1, & \sum_{i=1}^{m} w_{ij} x_i = \max_{1 \leq k \leq n} \left(\sum_{i=1}^{m} w_{ik} x_i \right), \\ 0, & \text{其他}. \end{cases}$$

当输入向量和 $W^{(j)} = (w_{1j}, w_{2j}, \cdots, w_{mj})^{\mathrm{T}}$, $j = 1, 2, \cdots, n$ 单位化以后,被选中神经元的权数向量 $W^{(j)}$ 同输入向量 X 的欧氏空间距离最近. 因此,它的学习规则是

$$w_{ij}(t+1) = w_{ij}(t) + \delta w_{ij}(t),$$

其中

$$\delta w_{ij}(t) = y_j(t)(x_i(t) - w_{ij}(t)).$$

竞争学习的最终效果使得相近的输入数据分为同一类. Kosko[11]用下面的一个简单神经网络例子"保持距离"来模拟分类问题. 神经网络见图 5.17.

输入 (x_1, x_2) 服从 $[-1, 1]$ 的均匀分布, 初始权数 $W^{(j)} = \begin{bmatrix} w_{1j} \\ w_{2j} \end{bmatrix}$ 在 $[-1,1] \times [-1,1]$ 随机选取. 由上面讨论的结果, $W^{(j)}$ 同输入向量 X 的欧氏空间距离越近, X 通过竞争被归为 j 类的可能性越大. 通过 24 982 次竞争学习后,再竞争学习的模拟输出结果如示意图 5.18,其中黑团部分表示对应的 $W^{(j)}$ 在以后竞争学习的变化区域.

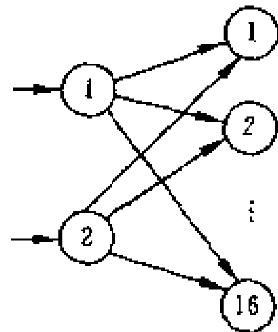


图 5.17 Kosko 神经网络

无监督学习的另一类应用是根据自适应谐振理论(ART)构成的神经网络,关心的读者可以参考文献[10].

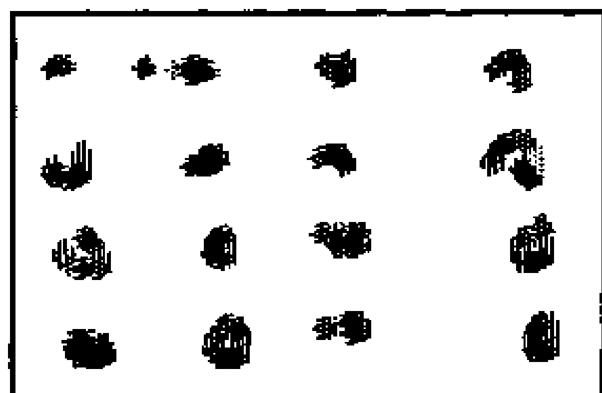


图 5.18　24 982 次竞争学习的模拟示意图

5.5　反馈型神经网络

反馈型神经网络(feed-back neural networks)的一般结构如图 5.19.

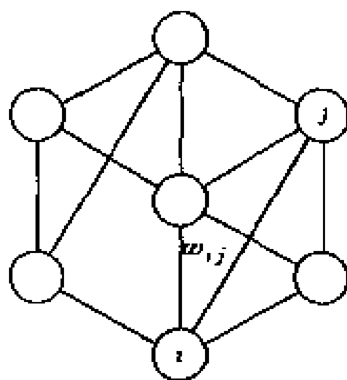


图 5.19　反馈型神经网络的一般结构

即神经元之间的信息传递关系不再是从一层到另一层，而是各神经元之间存在着联系. 反馈型神经网络有连续和离散系统两类，主要取决于用微分方程模型还是用差分模型描述.

5.5.1　连续系统神经网络

记 t 时刻第 i 个神经元接受到的信息量为 $Z_i(t)$，信息量的变化为

$$\frac{\mathrm{d}z_i(t)}{\mathrm{d}t} = -Az_i(t) + I_i(t) + \sum_{j=1}^{n} w_{ij} y_j(t),$$
$$y_i(t) = f_i(z_i(t)), \quad i = 1,2,\cdots,n, \tag{5.28}$$

其中，$y_i(t)$为 t 时刻第 i 个神经元的输出；$I_i(t)$为 t 时刻 i 神经元的外部传递信息；A 是一个正数，表示退化系数；w_{ij}表示两个神经元 i 和 j 相连的权数.(5.28)是一个动力系统方程.连续系统神经网络要求(5.28)式中函数 $z_i(t)$有很好的光滑性.下面给出动力系统的一些基本术语和概念.

动力系统微分方程组可以简单记

$$\frac{\mathrm{d}z(t)}{\mathrm{d}t} = g(z,t), \tag{5.29}$$

其中，$z(t)=(z_1(t),z_2(t),\cdots,z_n(t))^{\mathrm{T}}$.

定义 5.1　平衡点

如果

$$g(z_e,t) = 0, \forall\, t,$$

则称 z_e 是动力系统的平衡点，也称 z_e 为吸引子.

定义 5.2　稳定性

满足下列条件的平衡点 z_e 称为稳定点：

任给 $\epsilon>0$ 和 $t_0\geqslant 0$，存在 $\delta(\epsilon,t_0)>0$，当 $t\geqslant t_0$，$\|z_0-z_e\|<\delta(\epsilon,t_0)$时，有

$$\| z(t;z_0,t_0) - z_e \| < \epsilon,$$

其中，$z(t;z_0,t_0)$表示以 z_0 为起始点，t_0 为起始时间满足(5.29)的一条参数曲线.称系统(5.29)在 z_e 为稳定的.

定义 5.3　渐近稳定性

满足下列条件的稳定点 z_e 是渐近稳定点：

任给 $t_0 \geq 0$，存在 $\eta(t_0) > 0$，当 $\| z_0 - z_e \| < \eta(t_0)$ 时，有

$$\lim_{t \to \infty} z(t; z_0, t_0) = z_e.$$

称系统(5.29)在点 z_e 为渐近稳定的.

定义 5.4 吸引域

满足下列条件的点 z_0 组成的集合称为平衡点 z_e 的吸引域：

存在 $t_0 \geq 0$ 和 $\eta(t_0) > 0$，当 $\| z_0 - z_e \| < \eta(t_0)$ 时，有

$$\lim_{t \to \infty} z(t; z_0, t_0) = z_e.$$

反馈型神经网络的平衡点、平衡点的稳定性和吸引域性质的研究方法，主要是通过研究动力系统和与它对应的 Lyapunov 能量函数的性质. 连续动力系统与离散动力系统有很多相似之处[12]，因此连续动力系统是反馈型神经网研究的一个基础. 从物理学的角度有这样的现象：随着系统的运动，其储存的能量随时间的增长而衰减，直至趋于能量极小的平衡状态. 由此原理，反馈型神经网络研究一般采用稳定性的 Lyapunov 第二方法，或称直接法，它首先给出微分方程和对应的能量函数或广义能量函数，或称 Lyapunov 函数，在研究函数的特性后，可以不用求解系统的运动方程，而给出系统平衡稳定性的信息. 采用连续动力系统研究确定性神经网络的典型是 Hopfield 人工神经网络[5].

Hopfield 利用模拟电子线路功能构造了反馈型人工神经网络的电路模型，建立的能量函数表达式为

$$E(y) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{i=1}^{n} \frac{1}{R_i}\int_0^{y_i} f_i^{-1}(x)\mathrm{d}x - \sum_{i=1}^{n} I_i y_i, \tag{5.30}$$

其中，$y_i = f_i(z_i)$，f_i 为 Sigmoid 函数，w_{ij} 为 i 和 j 两个神经元的权数，电路中连线的电导具有对称性 $w_{ij} = w_{ji}$，R_i 是模拟电子线路电阻的大小，z_i 为第 i 个神经元的接受值，I_i 为外部偏置电流输入

值，$\sum_{i=1}^{n}\frac{1}{R_i}\int_0^{y_i}f_i^{-1}(x)\mathrm{d}x$ 称为增益项.

(5.30)的能量函数对应一个动力系统

$$\begin{cases}\dfrac{\mathrm{d}z_i}{\mathrm{d}t}=-Az_i+\sum_{j=1}^{n}w_{ij}y_j+I_i,\\ y_i=f_i(z_i),\quad i=1,2,\cdots,n,\end{cases}\tag{5.31}$$

其中 A 是与 R_i 有关的一个常数. 当 $R_i=R>0$ 时，$A=1/R$.

(5.30)与(5.31)有如下关系

$$-\frac{\mathrm{d}z_i}{\mathrm{d}t}=\frac{\partial E}{\partial y_i}.\tag{5.32}$$

由(5.30)、(5.31)和(5.32)，得到

$$\begin{aligned}\frac{\mathrm{d}E}{\mathrm{d}t}&=\sum_{i=1}^{n}\frac{\partial E}{\partial y_i}\frac{\mathrm{d}y_i}{\mathrm{d}t}=-\sum_{i=1}^{n}\frac{\mathrm{d}z_i}{\mathrm{d}t}\frac{\mathrm{d}y_i}{\mathrm{d}t}=-\sum_{i=1}^{n}\frac{\mathrm{d}z_i}{\mathrm{d}y_i}\left(\frac{\mathrm{d}y_i}{\mathrm{d}t}\right)^2\\&=-\sum_{i=1}^{n}\frac{\mathrm{d}f_i^{-1}(y_i)}{\mathrm{d}y_i}\left(\frac{\mathrm{d}y_i}{\mathrm{d}t}\right)^2\leqslant 0.\end{aligned}\tag{5.33}$$

特别当$\frac{\mathrm{d}z_i}{\mathrm{d}t}=0$，($i=1,2,\cdots,n$)时，$\frac{\mathrm{d}E}{\mathrm{d}t}=0$.

定理 5.2 若(5.30)是正定能量函数(即对$\forall\ y\neq 0$，有 $E(y)>0$)，且满足$\frac{\mathrm{d}E(y)}{\mathrm{d}t}\leqslant 0$. 则系统(5.31)的任何一个平衡点是稳定的.

证明可以参见文献[13].

Lyapunov 第二方法的基本思想是：先给以一个动力系统，而后构造对应的 Lyapunov 函数，通过对 Lyapunov 函数的研究来确定动力系统的稳定性. 一个系统的 Lyapunov 函数不具有唯一性.

Lyapunov 函数的最简单形式为二次型

$$E(\boldsymbol{y})=\boldsymbol{y}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{y},$$

其中 $\boldsymbol{B}$ 为实对称阵. 而矩阵 $\boldsymbol{B}$ 的元素有待确定. 如(5.30)可以写成

$$E = Y^{\mathrm{T}}BY + \sum_{i=1}^{n} \frac{1}{R_i} \int_0^{y_i} f_i^{-1}(x)\mathrm{d}x - \sum_{i=1}^{n} I_i y_i,$$

是一个二次型，其中，

$$B = -\frac{1}{2}(w_{ij})_{n\times n},\ Y = (y_1, y_2, \cdots, y_n)^{\mathrm{T}}.$$

由定理 5.2，要使(5.30)正定的一个充分条件为 B 矩阵是正定阵.

Lyapunov 函数方法研究的是每一点及邻域的变化情况，除非函数有非常好的特性，一般情况下所研究的结论都是局部性的.

从(5.30)、(5.31)、(5.32)和(5.33)的推导看出，Hopfield 神经网络模型正是 Lyapunov 第二方法的应用. 第一步是根据实际问题构造能量函数，然后利用电路系统的背景建立(5.31)的动力系统方程，其次再利用动力系统方程的计算，求解平衡点. (5.33)表明，随着时间的推移，能量函数逐渐下降. 当(5.31)的系统方程组达到平衡点时，(5.33)也降到了一个能量函数的导数为 0 点. 网络达稳定状态时，能量 E 为极小值. 于是 Hopfield 神经网络在组合优化问题中的计算归结为下面的计算步骤.

Hopfield 神经网络的计算步骤

STEP1 针对实际的组合优化问题构造能量函数，使得能量函数有好的稳定性，如具有正定二次型形式；

STEP2 由能量函数，根据(5.32)的关系求解出动力系统方程(5.31)；

STEP3 用数值计算的方法求解动力系统方程(5.31)并解出平衡点，具体的计算求解方法可以参见常用的软件，如 Mathlab 5.0，由定理 5.3 知平衡点为稳定点，达到极小值.

Hopfield 通过一个电路找到了人工神经网络的微分方程(5.31)和能量函数(5.30)的对应关系. 寻找一个微分方程的 Lyanupov 能量函数并不容易，这也因此成为很多理论工作者研究的一个方向. 就对(5.31)的微分系统，当一些参数条件变化时，研究它的稳定性仍然对实际应用有重要的指导意义. 文[14]研究了连接权不具有对称性和外加偏置电流 I_i 与时间参数 t 有周期输入的人工神经网络，得到该网络周期吸引子的存在性.

5.5.2 Hopfield 人工神经网络在 TSP 中的应用

已经有两种方法表示 TSP 问题的解(见第一章例 1.2 和例 1.8)，为了方便神经网络的求解，本节用第三种方法，一个 $n\times n$ 矩阵表示 TSP 的一个解，第 i 行表示商人到达该城市的顺序，第 i 行由 0,1 数字组成一个 n 维向量，其中只能有一个分量为 1，1 所在的序数表示商人到达的序数. 如向量(00010)表示商人第四个访问该城市.

例 5.4 四个城市 TSP 的一个解的矩阵表示为

	1	2	3	4
城市 1	0	1	0	0
城市 2	0	0	1	0
城市 3	1	0	0	0
城市 4	0	0	0	1

它表示商人行走的城市顺序为：3—1—2—4. □

从上例中可以看出，第三种解的表示方法采用的是例 1.2 的思想，只是附加了每一行的和每一列的和各为 1. 由此构造一个有 $n\times n$ 个神经元的神经网络. 对应 (i,j) 位置的神经元，其输出为 y_{ij}，输入为 z_{ij}，两个城市 x 和 y 的距离是 d_{xy}. 由期望每一行的和

为 1,一个等价的结论是

$$E_1 = \sum_{u=1}^{n} \sum_{i=1}^{n} \sum_{j \neq i} y_{ui} y_{uj} \tag{5.34}$$

为零.(5.34)中 $E_1=0$ 的含义是每一行最多有一个 1,可以全为 0.同理要求神经元的每一列的关系式

$$E_2 = \sum_{i=1}^{n} \sum_{u=1}^{n} \sum_{v \neq u} y_{ui} y_{vi} \tag{5.35}$$

为零.

要保证每一行每一列正好有一个 1,理想状态是

$$E_3 = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} y_{ij} - n \right)^2 \tag{5.36}$$

为零.

当 E_1,E_2,E_3 三项都达到理想状态也只保证得到 TSP 的一个可行解,而 TSP 的一个重要目标是得到一条费用最小的最短路程.于是,当 $d_{uv}=d_{vu}$时,考虑使下列形式

$$E_4 = \sum_{u=1}^{n} \sum_{v \neq u} \sum_{i=1}^{n} d_{uv} y_{ui} (y_{vi-1} + y_{vi+1}), \tag{5.37}$$

最小,其中 $y_{u0}=y_{un},y_{un+1}=y_{u1}$.由神经元的构造(参考例 5.4),两相邻的列表示商人行走的相邻城市,故 $d_{uv}y_{ui}y_{vi+1}$表示城市 u 和 v 之间的距离.(5.37)是其对称形式.

最后将能量函数写成

$$E = \frac{A}{2}E_1 + \frac{B}{2}E_2 + \frac{C}{2}E_3 + \frac{D}{2}E_4 + \alpha E_5, \tag{5.38}$$

A,B,C,D,α 为非负常数,表示 E_1,E_2,E_3,E_4,E_5 项对能量 E 的贡献大小.按(5.30)有

$$E_5 = \sum_{i,j} \frac{1}{R_{ij}} \int_0^{y_{ij}} f_{ij}^{-1}(x) \mathrm{d}x.$$

由(5.32)的关系,

$$
\begin{cases}
\dfrac{\mathrm{d}z_{ui}}{\mathrm{d}t} = -\dfrac{\partial E}{\partial y_{ui}} \\
\qquad = -\alpha z_{ui} - A\sum\limits_{j\neq i} y_{uj} - B\sum\limits_{v\neq u} y_{vi} \\
\qquad\quad - C\left(\sum\limits_{v=1}^{n}\sum\limits_{j=1}^{n} y_{vj} - n\right) - D\sum\limits_{v\neq u} d_{uv}(y_{v,i+1} + y_{v,i-1}), \\
y_{ui} = f(z_{ui}),
\end{cases}
\tag{5.39}
$$

(5.39)的系数整理后

$$
\begin{cases}
w_{ui,vj} = -A\delta_{uv}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{uv}) \\
\qquad\quad - C - Dd_{uv}(\delta_{j,i+1} + \delta_{j,i-1}), \\
I_{ui} = nC,
\end{cases}
\tag{5.40}
$$

其中,

$$
\delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \qquad d_{uu} = 0.
$$

于是,(5.39)写成了(5.31)的标准形式.

上面的讨论将求解 TSP 问题归结为 Hopfield 计算常规步骤.由(5.38)的构造、(5.33)的推导和定理 5.3,知(5.39)的平衡点是稳定点.存在的问题是参数 α, A, B, C, D 的选取.这些参数的选择是实际计算中的难点之一.D 对应 TSP 问题的目标函数项,应该突出,但同时希望罚值 A、B 和 C 尽可能的大以便求到可行解.实际计算中常常采取多组参数计算比较,从中选择好的解.下例是 Hopfield 给出的神经元初始值和参数选取的规则.

例 5.5　Hopfield[4]在求解 10 个城市的 TSP 问题时,数值计算的参数选取如下:

$$
\alpha = 1, A = B = D = 500, C = 200.
$$

Sigmoid 函数为

$$y_{ui} = f(z_{ui}) = \frac{1}{1 + e^{-\frac{2z_{ui}}{\mu_0}}},$$

其中,$\mu_0 = 0.02$.

初始设置 y_{ui}都相等,使得

$$\sum_{u=1}^{10} \sum_{i=1}^{10} y_{ui} = 10,$$

即期望最终的输出结果使得每一行和每一列只有一个输出非常接近 1. 用初始相等的 y_{ui}可以解出 $z_{00} = -\frac{\mu_0}{2}\ln 9$. 给 z_{00}一个扰动

$$z_{ui} = z_{00} + \delta z_{ui}, \; -0.1\mu_0 \leqslant \delta z_{ui} \leqslant 0.1\mu_0,$$

其中,δz_{ui}为$[-0.1\mu_0, 0.1\mu_0]$均匀分布,为动力系统数值计算的初始点. □

Hopfield 的开创性工作是给出求解组合优化问题的一种神经网络方法,但其不足也是显而易见的. 第一是参数 A,B,C,D,α 的选取,过多的参数增加了计算的工作量. 第二是例 5.5 中参数 μ_0 的确定,μ_0 较小时,使得 S 形曲线有较好的识别力,输出值接近 0 或 1,但初始解的范围较小,因此计算收敛的速度慢. 当参数 μ_0 较大时,又难以使输出值接近 0 或 1,使得路径的描述产生混淆. 第三个不足是它的计算量和收敛的效果. 由于它的能量函数(5.38)实际上是罚函数. 罚函数势必造成计算量的增加. 再者是 A,B,C,D,α 五个参数选取难以突出 TSP 的目标值,过多突出 D 会加大解的不可行性,加大其他参数会忽略目标值.

根据 Sigmoid 函数的特性,当 μ_0 较小时或 y_{ij}接近 0 或 1 时,y_{ij}的变化对(5.38)的总能量影响不是很大,于是常见作者省略 E_5 这一项,省略后(5.38)变为

$$E = \frac{A}{2}E_1 + \frac{B}{2}E_2 + \frac{C}{2}E_3 + \frac{D}{2}E_4. \qquad (5.38')$$

用神经网络求解 TSP 问题还有其他形式的能量函数,如用

$y_{ijk}=1$ 表示商人的第 k 次行走是从 i 城市到 j 城市，则能量函数：

$$E=\frac{A}{2}\sum_{j=1}^{n}\Big(\sum_{i=1}^{n}\sum_{k=1}^{n}y_{ijk}-1\Big)^2+\frac{B}{2}\sum_{i=1}^{n}\Big(\sum_{j=1}^{n}\sum_{k=1}^{n}y_{ijk}-1\Big)^2$$

$$+\frac{C}{2}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}y_{ijk}-n\Big)^2+\frac{D}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\mathrm{d}_{ij}y_{ijk},$$

其中上式右端第 1 项表示城市 j 恰好进去一次，右端第 2 项表示城市 i 恰好出去一次，第 3 项表示只有一条路径，第 4 项为行走路线最短.

5.5.3 离散系统神经网络

离散系统神经网络的输出值一般有如下关系：

$$Y(t+1)=g(Y(t),t),\quad t\in I=\{i\,|\,i\geqslant t_0,i\text{ 为整数}\},\tag{5.41}$$

其中，$Y(t)=(y_1(t),y_2(t),\cdots,y_n(t))^{\mathrm{T}}$. 如采用符号函数的反馈神经网络，其关系为

$$y_i(t+1)=\operatorname{sgn}\Big(\sum_{j=1}^{n}w_{ij}y_j(t)-\theta_i\Big),\quad i=1,2,\cdots,n,\tag{5.42}$$

记

$$z_i(t+1)=\sum_{j=1}^{n}w_{ij}y_j(t)-\theta_i,\quad i=1,2,\cdots,n.\tag{5.43}$$

连续动力系统中，定义 5.1 的平衡点 z_e 与参数 t 无关，也就使得输出 Y 与 t 无关. 于是，在离散动力系统中，可以用输出值 Y 的特性定义平衡点.

$$Y_e=g(Y_e,t),\ \forall\, t\in I,$$

则称 Y_e 为离散系统的平衡状态.

类似定义 5.2、定义 5.3 和定义 5.4，只需将原定义中的时间参数 t 看成离散的就可以定义离散系统的稳定性、渐近稳定性和

吸引域等概念.

在离散系统的神经网络中,当采用符号激活函数时,Hopfield建立的能量函数(5.30)变为

$$E(y) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} y_i y_j + \sum_{i=1}^{n} \theta_i y_i, \tag{5.44}$$

其中,$y_i = f_i(z_i)$,f_i 为符号函数,$w_{ij} = w_{ji}$ 为 i 和 j 神经元的权数,z_i 为第 i 个神经元的接受值,I_i 为外部输入值. 因为(5.30)中的 $\sum_{i=1}^{n} \frac{1}{R_i} \int_0^{y_i} f_i^{-1}(x)\mathrm{d}x = 0$,因此删去这一项.

由(5.43),得到能量函数(5.44)对应的一个动力系统为

$$\begin{cases} z_i(t+1) - z_i(t) = -z_i(t) + \sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i, \\ y_i = f_i(z_i), \\ i = 1,2,\cdots,n. \end{cases} \tag{5.45}$$

(5.45)的动力系统方程转化为

$$y_i(t+1) = f\Big(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\Big), \quad i = 1,2,\cdots,n,$$

其中 f 如(5.42)的符号函数. 本节仅以符号函数研究它的稳定性,故以(5.42)进行研究.

从算法的构造上,可以采用同步和异步两种形式. 所谓异步算法,是指每次只调整一个神经元,可表达为

$$\begin{cases} y_i(t+1) = \mathrm{sgn}\Big(\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\Big), \\ y_j(t+1) = y_j(t), \quad j \neq i, \end{cases} \tag{5.46}$$

它的含义是:在一步迭代中,只有第 i 个神经元发生变化,而其他神经元保持不变. 每次被调整的神经元 i 随机选定.

同步算法指同一时刻对所有神经元同时调整,表达式为

$$y_i(t+1)=\operatorname{sgn}\left(\sum_{j=1}^{n}w_{ij}y_j(t)-\theta_i\right),\quad i=1,2,\cdots,n.$$

例 5.6　三个神经元的反馈网络的权数为

$$\boldsymbol{W}=\begin{bmatrix}0&1&2\\1&0&3\\2&3&0\end{bmatrix},$$

在 t 时刻，一组输出 $\boldsymbol{Y}(t)=(1,0,1)^{\mathrm{T}}$，每个神经元的阈值都为 3. 在第 $t+1$ 时刻，同步算法使得每个神经元接受到的值为

$$\boldsymbol{Z}(t+1)=\boldsymbol{W}\boldsymbol{Y}(t)-\boldsymbol{\theta}=(2,4,2)^{\mathrm{T}}-(3,3,3)^{\mathrm{T}},$$

同步算法在 $t+1$ 时刻的输出为

$$\boldsymbol{Y}(t+1)=f(\boldsymbol{Z}(t+1))=(0,1,0)^{\mathrm{T}}.$$

异步算法的计算为：当 $t+1$ 步迭代选择第一个神经元时，它的接受值为

$$z_1(t+1)=(0,1,2)\boldsymbol{Y}(t)-\theta_1=-1,$$

所以，$t+1$ 时刻神经元的输出为

$$y_1(t+1)=0,\ y_2(t+1)=0,\ y_3(t+1)=1,$$

$$\boldsymbol{Y}(t+1)=(0,0,1)^{\mathrm{T}}.$$

若 $t+2$ 步迭代选择第二个神经元变化时，

$$z_2(t+2)=(1,0,3)\boldsymbol{Y}(t+1)-\theta_2=(1,0,3)\begin{bmatrix}0\\0\\1\end{bmatrix}-3=0,$$

所以

$$y_1(t+2)=0,\ y_2(t+2)=0,\ y_3(t+2)=1,$$

$$\boldsymbol{Y}(t+2)=(0,0,1)^{\mathrm{T}}.$$

若 $t+3$ 步迭代选择第三个神经元变化时，

$$z_3(t+3)=(2,3,0)\boldsymbol{Y}(t+2)-\theta_3=(2,3,0)\begin{bmatrix}0\\0\\1\end{bmatrix}-3=-3,$$

所以

$$y_1(t+3)=0,\ y_2(t+3)=0,\ y_3(t+3)=0,$$
$$\boldsymbol{Y}(t+3)=(0,0,0)^T.$$

虽说异步算法采用上面例中相邻的三次迭代达到同步算法对三个神经元进行的计算，但输出结果(0,0,0)与同步算法的输出结果(0,1,0)的差别是显而易见的. □

从例 5.6 可以看出同步算法与异步算法的区别.

由于同步算法的变化同连续系统的变化规律相同，因此同步算法的稳定性与连续系统的稳定性(定理 5.2)有相同的结论.

定理 5.3 若(5.44)中的权数 $\boldsymbol{W}$ 是正定矩阵，采用符号函数的同步算法的平衡点是系统(5.45)的稳定点.

证明 记(5.44)

$$E(t)=-\frac{1}{2}\boldsymbol{Y}^{\mathrm{T}}(t)\boldsymbol{W}\boldsymbol{Y}(t)+\boldsymbol{Y}^{\mathrm{T}}(t)\boldsymbol{\theta},$$

$$\Delta E(t)=E(t+1)-E(t),$$

$$\Delta\boldsymbol{Y}(t)=\boldsymbol{Y}(t+1)-\boldsymbol{Y}(t)=(\Delta y_1(t),\Delta y_2(t),\cdots,\Delta y_n(t))^{\mathrm{T}},$$

于是，

$$\Delta E(t)=-\Delta\boldsymbol{Y}^{\mathrm{T}}(t)(\boldsymbol{W}\boldsymbol{Y}(t)-\boldsymbol{\theta})-\frac{1}{2}\Delta\boldsymbol{Y}^{\mathrm{T}}(t)\boldsymbol{W}\Delta\boldsymbol{Y}(t).\quad(5.47)$$

令

$$\boldsymbol{Z}(t+1)=\boldsymbol{W}\boldsymbol{Y}(t)-\boldsymbol{\theta}=(z_1(t+1),z_2(t+1),\cdots,z_n(t+1))^{\mathrm{T}}.$$

当 $z_i(t+1)>0$ 时，

$$y_i(t+1)=f(z_i(t+1))=1,$$

得 $\Delta y_i(t)\geqslant 0$，推出 $\Delta y_i(t)z_i(t+1)\geqslant 0$.

当 $z_i(t+1)\leqslant 0$ 时，

$$y_i(t+1)=f(z_i(t+1))=0,$$

得 $\Delta y_i(t)\leqslant 0$，推出 $\Delta y_i(t)z_i(t+1)\geqslant 0$.

综合上面的讨论，

$$- \Delta Y^T(t)(WY(t) - \theta) \leqslant 0.$$

由定理中 W 为正定的条件，当 $\Delta Y(t) \neq 0$ 时，

$$\Delta E(t) = -\Delta Y^T(t)(WY(t) - \theta) - \frac{1}{2}\Delta Y^T(t) W \Delta Y(t) < 0.$$

因此，由定理 5.3，任何一个平衡状态为稳定的. □

离散系统的异步算法有别于连续系统的神经网络，我们在此专门讨论. 在以下的讨论中，特假定下列条件满足：

① W 是对称的 n 阶方阵，且 $w_{ii} \geqslant 0$；

② $y_i(t) \in \{0,1\}$；

③ 符号函数为

$$\mathrm{sgn}(x) = \begin{cases} 1, & x \geqslant 0, \\ 0, & \text{其它}, \end{cases}$$

此处的符号函数有别于前面的函数定义，它把 $x=0$ 的点赋予值 1.

定理 5.4　若能量函数(5.44)有下界，则对任意的初始状态，满足上面假设的异步 Hopfield 算法，最终收敛到一个平衡状态.

证明　按异步算法任选一个神经元 i，变化

$$\Delta Y(t) = (0, \cdots, 0, \Delta y_i(t), 0, \cdots, 0)^T,$$

于是，(5.47)等价于

$$\begin{aligned} \Delta E(t) &= -\Delta y_i(t)\Big[\sum_{j=1}^{n} w_{ij} y_j(t) - \theta_i\Big] - \frac{1}{2} w_{ii} \Delta y_i^2(t) \\ &= -\Delta y_i(t)\Big[z_i(t+1) + \frac{1}{2} w_{ii} \Delta y_i(t)\Big]. \end{aligned} \tag{5.48}$$

按 $\Delta y_i(t)$ 的取值分下列情况讨论：

CASE1　$\Delta y_i(t)=0$，即 $y_i(t+1)=y_i(t)$；

CASE2　$\Delta y_i(t)=1$，即 $y_i(t+1)=1$，$y_i(t)=0$；

CASE3　$\Delta y_i(t)=-1$，即 $y_i(t+1)=0$，$y_i(t)=1$.

下面按上面的三种情况分别讨论.

CASE1 $\Delta y_i(t)=0$ 推出 $\Delta E(t)=0$.

CASE2 由 $\Delta y_i(t)=1$,

$y_i(t+1)=1$, $y_i(t)=0$, $y_i(t+1)=\mathrm{sgn}(z_i(t+1))$,

得到

$$z_i(t+1)\geqslant 0.$$

再由 $w_{ii}\geqslant 0$ 得 $\Delta E(t)\leqslant 0$.

CASE3 $\Delta y_i(t)=-1$ 时,$y_i(t+1)=0$, $y_i(t)=1$,由上面新给的有关符号函数的假设,推出 $z_i(t+1)<0$ 及 $\Delta E(t)<0$.

采用异步算法,由 $E(t)$的非增性,经过一定的迭代后,将达到 $E(t)$不再下降.此时有 $\Delta E(t)=0$.若满足 $\Delta y_i(t)=0,\forall\, i$,则 $Y(t)$ 为一个平衡点.但条件 $\Delta y_i(t)=0,\forall\, i$ 不满足时,由上面的三种情况讨论,存在 i,使得 $\Delta y_i(t)>0$ 且

$$z_i(t+1)+\frac{1}{2}w_{ii}\Delta y_i(t)=0.$$

由此而得到

$$z_i(t+1)=0,\ w_{ii}=0.$$

此时记

$$H=\{i\,|\,y_i(t+1)=1, y_i(t)=0, z_i(t+1)=0, w_{ii}=0\},$$

则对 H 中的任何一个神经元再迭代一次,就达到

$$y_i(t+1)=y_i(t+2)=1.$$

对于已经满足平衡的集合 $J=\{j\,|\,\Delta y_j(t)=0\}$,经过上面的一次迭代后,再对 J 中的神经元迭代(第 τ 次)是否会出现不平衡?不平衡的情况为

① $y_i(t)=y_i(t+1)=0$, $y_i(\tau)=1$, $y_i(\tau+1)=0$;

② $y_i(t)=y_i(t+1)=0$, $y_i(\tau)=0$, $y_i(\tau+1)=1$;

③ $y_i(t)=y_i(t+1)=1$, $y_i(\tau)=0$, $y_i(\tau+1)=1$;

④ $y_i(t)=y_i(t+1)=1$, $y_i(\tau)=1$, $y_i(\tau+1)=0$.

由 CASE1、2、3,每次迭代中的能量函数都是非升的结论和此

时能量函数对任何神经元迭代不再下降，则知①和④是不可能发生的. 而②和③又归于 H 中的情形. 因神经元个数有限，最后一定达到全部平衡. □

由离散系统所具有的特性，类似定义 5.2 和定义 5.3，即可定义离散系统的稳定性和渐近稳定性.

定义 5.5　$Y\in\{0,1\}^n$ 的邻域为

$$N(Y)=\{Y+\delta_i e_i \mid i=1,2,\cdots,n\},$$

其中，e_i 表示第 i 个分量为 1 其他分量为 0 的 n 维向量，

$$\delta_i=\begin{cases}-1, & 当\ y_i=1,\\ 1, & 当\ y_i=0,\end{cases}$$

若能量函数在 Y^* 满足

$$E(Y^*)\leqslant E(Y),\ \forall\ Y\in N(Y^*),$$

则称 Y^* 为邻域 $N(Y^*)$ 的能量极小点，简称能量极小点.

离散系统的邻域同组合优化问题的一类邻域定义相同(第 1 章 1.3 节)，因此可以相似地给出离散系统的其他邻域定义.

定理 5.5　若 $z_i(t+1)=0$ 时，定义 $y_i(t+1)=y_i(t)$，则每个能量极小点为平衡点.

证明　记能量极小点为 Y^*

$$z_i^*=\sum_{j=1}^{n}w_{ij}y_j^*-\theta_i,\ y_i(2)=f(z_i^*),\ \Delta y_i^*=y_i(2)-y_i^*.$$

若为平衡点，则结论成立. 当某一个神经元 i 不平衡时，记神经元的能量变化为 ΔE_i，则由(5.48)有

$$\begin{aligned}\Delta E_i&=-\Delta y_i^*\left[\sum_{j=1}^{n}w_{ij}y_j^*-\theta_i+\frac{1}{2}w_{ii}\Delta y_i^*\right]\\&=-\delta_i\left[z_i^*+\frac{1}{2}w_{ij}\delta_i\right].\end{aligned}$$

按定理 5.4 证明中讨论的三种情况和 $z_i(t+1)=0$ 时，定义 $y_i(t+1)=y_i(t)$，不平衡的可能有

CASE1 $\delta_i=1$ 时，$y_i^*=0$，则

$$z_i^* + \frac{1}{2}w_{ii} \leqslant 0,$$

由 $w_{ii}\geqslant 0$ 的条件，推出 $z_i^* \leqslant 0$，进而推出

$$y_i(2) = f(z_i^*) = \begin{cases} 0, & \text{当 } z_i^* < 0, \\ y_i^*, & \text{当 } z_i^* = 0. \end{cases}$$

于是，$y_i(2)=y_i^*$，与不平衡矛盾.

CASE2 $\delta_i=-1$ 时，$y_i^*=1$，则

$$z_i^* + \frac{1}{2}w_{ii}\delta_i \geqslant 0,\ z_i^* \geqslant \frac{1}{2}w_{ii} \geqslant 0,$$

进而推出

$$y_i(2) = f(z_i^*) = \begin{cases} 1, & \text{当 } z_i^* > 0, \\ y_i^*, & \text{当 } z_i^* = 0. \end{cases}$$

于是，$y_i(2)=y_i^*$，与不平衡矛盾.

综合 CASE1 和 CASE2 的讨论，得到 Y^* 是一个平衡点. □

虽说能量函数的极小点是一个平衡点，但是否是一个稳定点或是一个渐近稳定点？从极小点邻域中的一点出发，在什么样的条件下才能回到这个极小点？由于离散系统的特性，定义了如弱吸引等概念，由此，得到相关的结论，在此不介绍这些内容，感兴趣的读者可参考人工神经网络算法的专著，见文献[10].

定理 5.6 若离散系统神经网络的权矩阵对角线还满足diag$(\boldsymbol{W})=(0,0,\cdots,0)$，则网络的所有平衡点为能量函数极小点.

证明 记平衡点 $\boldsymbol{Y}^*=(y_1^*,y_2^*,\cdots,y_n^*)^{\mathrm{T}}$，从 $\boldsymbol{Y}^*$ 的邻域中任选一点，不妨设为第 i 个神经元变化，则由(5.48)能量变化为

$$\Delta E_i = -\delta_i\left[z_i^* + \frac{1}{2}w_{ii}\delta_i\right] = -\delta_i z_i^*.$$

当 $\delta_i=1$ 时，$y_i^*=0$，由平衡点的性质 $y_i(2)=f(z_i^*)=0$，知 $z_i^* \leqslant 0$. 由此得到 $\Delta E_i\geqslant 0$.

当 $\delta_i=-1$ 时，$y_i^*=1$，由平衡点的性质 $y_i(2)=f(z_i^*)=1$，知 $z_i^*\geqslant 0$. 由此得到 $\Delta E_i\geqslant 0$.

综合上面两点，平衡点为能量函数极小点. □

Hopfield 神经网络异步算法的基本步骤为

STEP1　任选一个初始状态 $Y(0)\in\{0,1\}^n$；

STEP2　随机选一个神经元，按(5.46)更新状态；

STEP3　检验 $Y(t)$ 是否为网络的平衡点，若是，转 STEP4；否则，转 STEP2；

STEP4　输出 $Y(t)$.

Hopfield 神经网络与一电路相对应，因此在硬件上有实现的可能，故有很好的发展前景. 该方法能量函数的建立相对简单，可以应用于组合优化问题及识别问题.

由于构造的能量函数有局部极值点，因此可能收敛到局部最优. 特别是能量函数中的参数选取，如 TSP 中的 A、B、C 和 D，决定收敛的速度、收敛的效果. 因此，选择参数时要求应用人员有很高的技术性和对问题有较深的了解. 在实际应用中，通常处理的方法是大量计算，以尝试找到比较好的参数搭配.

正因为存在局部极小点，因此采用数值计算的方法就无法避免落入局部极小. 于是相关的吸引子性质研究成为理论研究的热点. 在算法的实现上，为了避免局部极小，采用与模拟退火、遗传算法等相结合的方法.

5.5.4　变形算法——TSP 中的应用

变形算法(elastic net method)是反馈型神经网络在组合优化问题中的一种应用，它来源于 Kohonen[15] 的自组织影射模型. 在

这种算法中，组合优化问题的任一实例数据为网络的输入.在进化过程中，网络修改它的连接权.我们在此介绍变形模型的一个应用——TSP中的弹性网络[16]——来理解变形算法的基本原理.

假设TSP的n城市位置为$X_i=(x_{i1},x_{i2},\cdots,x_{im})^T$，$i=1,2,\cdots,n$，其中$m$为空间的维数.在这个空间中给以$N>n$个临时点$Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T$，$a=1,2,\cdots,N$.我们期望将临时点同城市位置匹配使得：每个城市仅同一个临时点匹配且这些临时点形成的闭路满足$\sum_a|Y_a-Y_{a+1}|$尽量小，其中$Y_{N+1}=Y_1$.若当城市i同临时点a匹配，则定义一个变量$s_{ia}=1$，否则$s_{ia}=0$.于是可以给出下列的能量函数

$$E(s_{ia},Y_a)=\frac{1}{2T}\sum_{ia}s_{ia}|X_i-Y_a|^2+\frac{\gamma}{2}\sum_a|Y_a-Y_{a+1}|^2. \tag{5.49}$$

(5.49)的第一项的功能是匹配，需要满足$\sum_{ia}s_{ia}=n$.当已经匹配且T渐近零的时候，(5.49)的第二项功能是使得这些临时点形成的闭路尽量小且点的分布尽量均匀.由于Y_a和s_{ia}是变量和s_{ia}求解的难度，用另一个效用能量函数

$$E_{eff}(Y_a)=-T\sum_i\log_{10}\Big(\sum_a e^{-|X_i-Y_a|^2/2T^2}\Big)+\frac{\gamma}{2}\sum_a|Y_a-Y_{a+1}|^2, \tag{5.50}$$

使其达到最小.

在(5.50)中，从右边的第一项看出，当每一个X_i都有一个临时点与其匹配时，

$$\sum_a e^{-|X_i-Y_a|^2/2T^2}=\sum_{a\in M}e^{-|X_i-Y_a|^2/2T^2}+\sum_{a\notin M}e^{-|X_i-Y_a|^2/2T^2}\to k,T\to 0,$$

其中，M表示同X_i匹配的临时点集合，k为同X_i匹配的临时点个数.此时，当T趋向零时，(5.50)右端第一项的值接近零.当存在

一个 X_i，没有一个临时点与其匹配时，由 $\sum_a e^{-|X_i-Y_a|/2T^2}\to 0, T\to 0$，推出(5.50)右端第一项随着 T 的趋向零而增加，因此需要继续优化.(5.50)右端的第二项要求 N 个临时点形成的闭路尽量小和尽可能在闭路上均匀分布.(5.50)中只有 Y_a 为决策变量.能量函数的梯度为

$$\frac{\partial E_{eff}(Y_a)}{\partial Y_a} = -\sum_i v_{ia}(X_i - Y_a)/T + \gamma(2Y_a - Y_{a+1} - Y_{a-1}),$$

其中，

$$v_{ia} = \frac{\exp\left\{-\frac{|X_i - Y_a|^2}{2T^2}\right\}}{\sum_b \exp\left\{-\frac{|X_i - Y_b|^2}{2T^2}\right\}}. \tag{5.51}$$

$E_{eff}(Y_a)$对变量 Y_a 的下降梯度为

$$\Delta Y_a = -\eta\frac{\partial E_{eff}(Y_a)}{\partial Y_a},\quad \eta\geqslant 0 \text{ 的参数}. \tag{5.52}$$

由于 $E_{eff}(Y_a)$是一个非线性函数，且具有光滑性.采用梯度下降方向的迭代算法收敛到局部最优要求 η 满足一定的条件，在此不进一步讨论，关心这一部分内容的读者可参考非线性规划的有关论著，如文献[17，第 2 章].具有变形算法在 TSP 中的应用——弹性算法的基本变化可以由图 5.20 描述.

图 5.20 描述了弹性算法计算时变化的四个阶段.图中"*"表示城市的位置.初始时在城市的重心位置的一个小圆周上，所有临时点均匀分布其上，如图 5.20(a)所示.经过若干次迭代，这些临时点向各城市靠近，就如同一个橡皮圈被拉向各城市一样，这种变化如图 5.20 中的(b)和(c)变化一样.最后，在理想情况下，每个城市都有临时点匹配且其他点按距离均匀分布在这些匹配点连成的线段上，就如同图 5.20(d)所示.

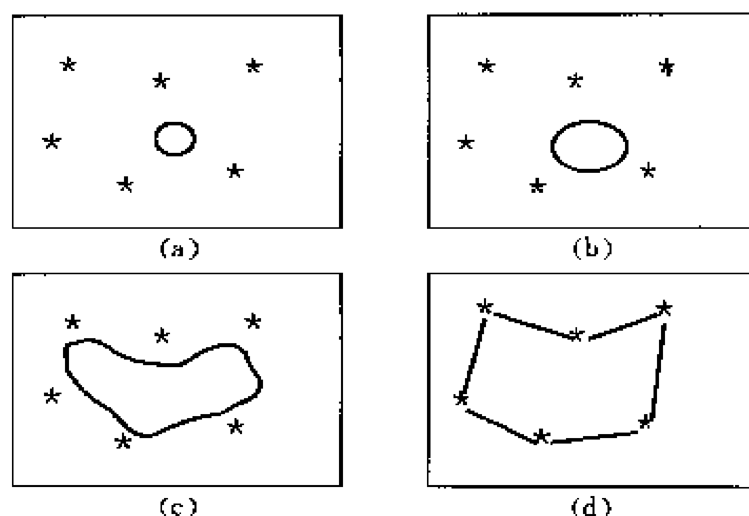


图 5.20 弹性算法的基本变化

TSP 的弹性算法如下

STEP1 给定 TSP 实例 n 个城市的坐标$\{X_i|i=1,2,\cdots,n\}$;

STEP2 选一个比 n 大得多的临时点数 N;

STEP3 选择合适的参数 γ、初始温度 T_0 和初始步长参数 η_0;

STEP4 求$\{X_i|i=1,2,\cdots,n\}$的重心;随机将 $Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T,a=1,2,\cdots,N$ 均匀地布在以重心为圆心的一个小圆周上;

STEP5 在算法没有达到停止条件前,重复以下计算:

a. 更新临时坐标 Y_a 为

$Y_a(T_l)=Y_a(T_{l-1})+\Delta Y_a(T_{l-1})$,

其中,$\Delta Y_a(T_{l-1})$由(5.52)计算;

b. $l:=l+1$; $T_l=\alpha T_{l-1}(0<\alpha\leqslant 1)$.

变形算法为什么可以认为是反馈型神经网络?第一是因为将$\{X_i|i=1,2,\cdots,n\}$和临时点看成神经元.第二是因为 v_{ia}可以看成

神经元 i 和 a 的权数，当问题的参数 $\{X_i|i=1,2,\cdots,n\}$ 输入后，这一权数由(5.51)随着 $Y_a=(y_{a1},y_{a2},\cdots,y_{am})^T, a=1,2,\cdots,N$ 的改进一步步地改进.

弹性网络法适合于欧氏距离为费用的 TSP 问题，Peterson[18] 将这一算法应用到 TSP 并对大量典型数据进行计算比较，发现此算法的效率较高且临时点的个数也较小 $N=2n$.

变形算法的特点是模拟橡皮圈被拉时的变化过程，分别建立了(5.50)和(5.52)的能量函数和下降梯度。在此介绍这个算法的目的是希望读者能将这一种思想在其他实际问题中得以应用.

练 习 题

1. 前向神经网络适合哪些组合优化问题？这些问题可否用反馈型神经网络处理？举例说明.

2. 反馈型神经网络适合求解什么样的组合优化问题？它在哪些方面比前向型神经网络有优越性？举例说明.

3. 按例 5.5 的各参数设定实现 Hopfield 神经网络算法在 TSP 的应用，是否能得到与文献[4]相类似的结果？

4. 比较 Hopfield 神经网络和变形算法在 TSP 的应用效果.

5. 就你关心或实际遇到的组合优化问题用神经网络方法求解. 试同其他诸如禁忌搜索、模拟退火、遗传算法等进行计算结果比较.

参 考 文 献

1. James W. Psychology(Briefer Course). New York: Holt, 1890

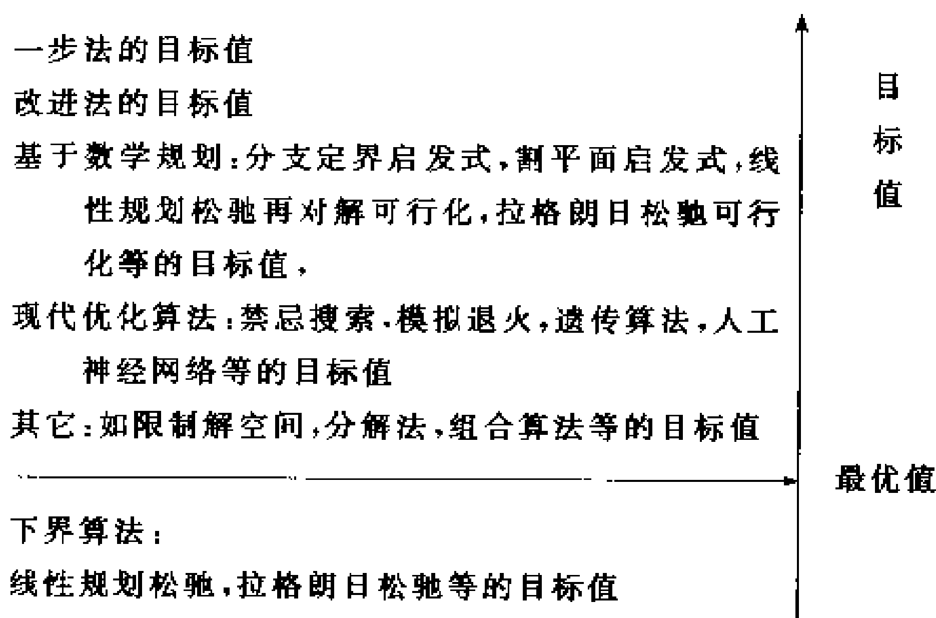
2. McCulloch W, Pitts W. A logical calculus of the ideas imminent in nervous activity. Bulletin of Mathematical Biophysics, 1943, 5:115～133

3. Minsky M, Papert S. Perceptrons. Cambridge, MA: MIT Press, 1969
4. Hopfield J, Tank D. 'Neural' computation of decisions in optimization problems. Biological Cybernetics, 1985, 52:141~152
5. Hopfield J, Tank D. Computing with neural circuits: a model. Science, 1986, 233:625~633
6. McClelland J, Rumelhart D. Explorations in Parallel Distributed Processing. Cambridge, MA: MIT Press, 1988
7. Widrow B, Stearns S. Adaptive Signal Processing. Englewood: Prentice Hall, 1985
8. Rumelhart D, Hinton G E, Williams R J. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge: MIT Press, 1986
9. Wu B. An introduction to neural networks and their applications in manufacturing. Journal of Intelligent Manufacturing, 1992, 3:391~403
10. 杨行峻,郑君里. 人工神经网络. 高等教育出版社, 1992
11. Kosko B. Neural Networks and Fuzzy Systems. Englewood: Prentice Hall, 1992
12. Hopfield J. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Science, USA, 1984, 81:3088~3092
13. 张芷芬等. 微分方程定性理论. 科学出版社, 1985
14. 李铁成,王铎. 一类带有周期输入的人工神经网络的渐近性质. 高校应用数学学报,1997, 12(Ser A): 25~28
15. Kohonen T. Self-Organization and Assocoative Memory. Springer, 1984
16. Durbin R, Willshaw D. An analogue approach to the TSP using an elastic net method. Nature, 1987, 326:689~691
17. 袁亚湘. 非线性规划数值方法. 上海科学技术出版社, 1993
18. Peterson C. Parallel distributed approaches to combinatorial optimization problems—— benchmark studies on TSP. Neural Computation, 1990, 2:261

第 6 章

拉格朗日松弛算法

当一个组合优化问题被判别为 NP-complete 或 NP-hard，目前解决这个问题的常用方法是构造启发式算法，求解尽量接近最优解的可行解. 这些算法包括第 2 章至第 5 章的局部搜索算法、禁忌搜索、模拟退火、遗传算法和人工神经网络等等. 以极小优化目标函数为例，这些算法给出的都是最优值的上界，第 1 章的 1.4.2 节给出这些算法的目标值同最优值关系的示意图如下：



评价一个算法好坏的一个标准是考察它所计算的目标值同最优目标值的差别. 由于组合优化问题的难度，求解最优值是非常困难的. 解决这个难点的一个有效方法是通过计算下界，用上界和下

界的差来评价算法.拉格朗日(Lagrange)松弛算法就是求解下界的一种方法.由于拉格朗日松弛算法的实现比较简单和有比较好的性质,它不仅可以用来评价算法的效果,同时可以用在其他算法中,以提高算法的效率.拉格朗日松弛算法包含两部分内容:一方面是提供下界,另一方面则演变为拉格朗日松弛启发式算法.

6.1 基于规划论的松弛方法

在此仅以整数规划为基础讨论,相应的结果可以推广到混合整数规划问题.整数规划的数学模型为

$$\begin{cases} z_{IP} = \min c^T x \\ \text{s.t.}\ Ax \geqslant b \\ x \in Z_+^n, \end{cases} \tag{6.1}$$

其中,决策变量 x 为 n 维列向量,c 为 n 维列向量,A 为 $m \times n$ 矩阵,b 为 m 维列向量;系数 c、A 和 b 取整数,Z_+^n 表示非负整数集合.

1. 线性规划松弛

在(6.1)中将整数变量约束松弛为实数,就可以得到

$$\begin{cases} z_{LP} = \min c^T x \\ \text{s.t.}\ Ax \geqslant b \\ x \in R_+^n, \end{cases} \tag{6.2}$$

其中 R_+^n 为非负实数集,称(6.2)为(6.1)的线性规划松弛.线性规划松弛扩大了整数规划的可行解区域.若记

$$S = \{x \in Z_+^n \mid Ax \geqslant b\},\ S' = \{x \in R_+^n \mid Ax \geqslant b\},$$

则有 $S \subseteq S'$,于是得到结论:

定理 6.1 $z_{LP} \leqslant z_{IP}$.

定理 6.1 说明线性规划松弛得到整数规划的一个下界.可以通过单纯形算法或多项式时间的内点算法[1],求得(6.2)的线性规

划的最优解.

当 S 中的一个解 x_0 满足 $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}_0=z_{\mathrm{LP}}$时，推出 x_0 为(6.1)的最优解.作为求解整数规划问题启发式算法的一部分，线性规划松弛适用于整数规划问题中决策变量是比较大的整数.对这样的问题，启发式算法的两阶段为：第一阶段将整数规划问题松弛为线性规划问题，求解线性规划问题的最优解；第二阶段将线性规划的最优解按四舍五入或类似的原则整数化，同时考虑解的可行性.

2. 对偶规划松弛方法

线性规划(6.2)的对偶形式为

$$\begin{cases} z_{\mathrm{DP}} = \max \boldsymbol{y}^{\mathrm{T}}\boldsymbol{b} \\ \text{s.t.}\ \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \leqslant \boldsymbol{c} \\ \boldsymbol{y} \in \mathbf{R}_{+}^{n}, \end{cases} \tag{6.3}$$

其中，决策 $\boldsymbol{y}$ 是一个 m 维列向量.

(6.2)和(6.3)都为线性规划问题，它们的计算方法相同，且由对偶理论得到 $z_{\mathrm{DP}}=z_{\mathrm{LP}}\leqslant z_{\mathrm{IP}}$. 至于采用(6.2)或(6.3)中哪一个求(6.1)的下界，则需比较哪一个计算简单.无论如何，单纯形算法和内点算法在实际应用中都可能因为耗时过大而不能满足要求.如在一个循环计算的算法中，每一次循环需要求解一个线性规划问题，当循环的步数较大时，无论用单纯形算法还是内点算法都会感到计算时间过多，由此，可能无法满足计算时间的要求.

3. 代理(surrogate)松弛法

当(6.1)的约束过多时，代理松弛法是通过一个约束

$$\sum_{j=1}^{n}\Big(\sum_{k=1}^{K}a_{i_k j}\Big)x_j \geqslant \sum_{k=1}^{K}b_{i_k}$$

替代(6.1)中的 K 个约束

$$\sum_{j=1}^{n}a_{i_k j}x_j \geqslant b_{i_k}, \quad k=1,2,\cdots,K.$$

极端的情况可以用一个约束

$$\sum_{j=1}^{n}\Big(\sum_{i=1}^{m}a_{ij}\Big)x_j \geqslant \sum_{i=1}^{m}b_i$$

松弛约束 $Ax \geqslant b$. 代理松弛法保证目标函数、整数变量约束不变，且因约束的减少造成计算量的减少.

4. 拉格朗日松弛方法

拉格朗日松弛方法的基本原理是：将造成问题难的约束吸收到目标函数中，并使得目标函数仍保持线性性，由此使得问题容易求解. 对它的研究产生兴趣主要基于下面的原因：

(1) 一些组合优化问题是 NP-hard，也就是在现有的约束条件下很难求得最优解. 但在原有的问题中减少一些约束后，求解问题的难度就大大的减少，达到在多项式时间内求得减少约束问题的最优解. 由此，将这些减少的约束称为难约束. 对于整数线性规划问题，将难约束吸收到目标函数后，问题又变得容易求解. 这时解的质量完全依赖于吸收到目标函数时所选取的参数.

例 6.1 集合覆盖问题(set covering problem)

设 $A=(a_{ij})_{m\times n}$，所有元素 $a_{ij}\in\{0,1\}$，且每一列对应一个费用 $c_j(j=1,2,\cdots,n)$. $a_{ij}=1$ 表示第 j 列覆盖第 i 行. 集合覆盖问题是以最小的费用选择一些列使得这些列覆盖所有的行. 它的数学模型为

$$z_{\mathrm{SC}} = \min\sum_{j=1}^{n}c_jx_j \tag{6.4}$$

$$\text{(SC)}\qquad \text{s.t.}\ \sum_{j=1}^{n}a_{ij}x_j \geqslant 1,\quad i=1,2,\cdots,m, \tag{6.5}$$

$$x_j\in\{0,1\},\quad j=1,2,\cdots,n. \tag{6.6}$$

集合覆盖问题是 NP-hard[2]. 若将(6.5)松弛，可得优化问题

$$z_{\mathrm{LRSC}}(\lambda) = \min\sum_{j=1}^{n}c_jx_j + \sum_{i=1}^{m}\lambda_i\Big(1-\sum_{j=1}^{n}a_{ij}x_j\Big)$$

$$\text{s.t.}\quad x_j\in\{0,1\},\quad j=1,2,\cdots,n,$$

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_m)^{\mathrm{T}} \geqslant 0.$$

记
$$d_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij},$$

则松弛后的模型为

$$\begin{aligned} z_{\mathrm{LRSC}}(\boldsymbol{\lambda}) &= \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ \text{s.t.} \quad & x_j \in \{0,1\}, \quad j = 1,2,\cdots,n, \\ & \boldsymbol{\lambda} \geqslant 0. \end{aligned} \tag{6.7}$$

(6.7)很容易求得最优解

$$x_j^* = \begin{cases} 1, & 若\ d_j \leqslant 0, \\ 0, & 其它. \end{cases}$$

最优值为

$$z_{\mathrm{LRSC}}(\boldsymbol{\lambda}) = \sum_{j=1}^{n} d_j x_j^* + \sum_{i=1}^{m} \lambda_i.$$

从松弛和求解的过程中看出：对给定的 $\boldsymbol{\lambda} \geqslant 0$，满足(6.5)和(6.6)的一个可行解自然满足(6.7)的约束，因此，$z_{\mathrm{LRSC}}(\boldsymbol{\lambda}) \leqslant z_{\mathrm{SC}}$ 且 $z_{\mathrm{LRSC}}(\boldsymbol{\lambda})$ 是 SC 问题的一个下界. 若(6.7)的一个可行解不满足约束(6.5)时，即存在 i，使得 $\sum_{j=1}^{n} a_{ij} x_j < 1$，可以通过调节 λ_i，使其增大而惩罚解的不可行性. 于是 $z_{\mathrm{LRSC}}(\boldsymbol{\lambda})$ 同 z_{SC} 的差距依赖于 $\boldsymbol{\lambda} \geqslant 0$ 的选取. 还可以看出松弛后的最优解非常容易得到，只需判别 d_j 的正负号. □

(2) 实际的计算结果证实拉格朗日松弛方法所给的下界相当不错，且计算时间可以接受. 同时，可以进一步利用拉格朗日松弛的基本原理构造基于拉格朗日松弛的启发式算法.

由上面四种松弛方法，可以给予松弛一个定义：

定义 6.1　问题

(RP)
$$z_{\mathrm{R}} = \min_{x \in S_{\mathrm{R}}} z_{\mathrm{R}}(\boldsymbol{x})$$

满足下列两条性质时，RP 称为整数规划(6.1)的一个松弛(relaxation)：

(1) 可行解区域兼容：$S\subseteq S_R$；

(2) 目标函数兼容：$c^T x\geqslant z_R(x),\forall\ x\in S$

其中，S_R 表示一个解集合，$z_R(x)$为实函数.

定理 6.2 若 RP 无可行解，则(6.1)也无可行解；若(6.1)有可行解，则 $z_{IP}\geqslant z_R$.

证明 当 RP 无可行解时，由可行解区域兼容性，$S=\varnothing$. 当(6.1)可行时，(6.1)的最优解为 RP 的一个可行解，所以 $z_{IP}=z_R$.

□

6.2 拉格朗日松弛方法的理论

理论告示我们，如果一个整数规划问题可以在多项式时间内求得最优解，没有必要用更复杂的算法去求解. 当面对一个 NP-hard 的整数规划问题时，除非 P=NP，构造多项式时间的最优算法已不可能. 本章是在整数规划问题为 NP-hard 的前提下讨论它的松弛方法. 为了适合拉格朗日松弛方法的讨论，将整数规划问题 IP 描述为

$$
\begin{aligned}
& z_{IP} = \min\ c^T x \\
(\text{IP}) \qquad & \text{s.t.}\ Ax \geqslant b(\text{复杂约束}), \\
& Bx \geqslant d(\text{简单约束}), \\
& x \in Z_+^n,
\end{aligned}
$$

其中，(A,b)为 $m\times(n+1)$整数矩阵，(B,d)为 $l\times(n+1)$整数矩阵. 记 IP 的可行解区域为

$$S = \{x \in Z_+^n\ |Ax \geqslant b, Bx \geqslant d\}.$$

在 IP 模型中，$Ax\geqslant b$ 为复杂约束的名称来自于：如果将该项约束去掉，则 IP 可以在多项式时间求到最优解，即假定

$$\min c^{\mathrm{T}} x$$
$$\text{s.t. } Bx \geqslant d \text{（简单约束）}, \tag{6.8}$$
$$x \in Z_{+}^{n},$$

可在多项式时间内求得最优解.

对给定的 $\lambda=(\lambda_1,\lambda_2,\cdots,\lambda_m)^{\mathrm{T}}\geqslant 0$，IP 对 λ 的拉格朗日松弛（在不对 λ 的取值产生混淆时，简称为 LR）定义为：

$$z_{\mathrm{LR}}(\lambda)=\min c^{\mathrm{T}}x+\lambda^{\mathrm{T}}(b-Ax)$$
$$\text{(LR)}\qquad \text{s.t. } Bx\geqslant d,$$
$$x\in Z_{+}^{n}.$$

LR 的可行解区域记为 $S_{\mathrm{LR}}=\{x\in Z_{+}^{n} \mid Bx\geqslant d\}$.

定理 6.3　LR 同(6.8)有相同的复杂性，且若 IP 的可行解区域非空，则

$$\forall\ \lambda\geqslant 0\Rightarrow z_{\mathrm{LR}}(\lambda)\leqslant z_{\mathrm{IP}}.$$

证明　令

$$g(x,\lambda)=c^{\mathrm{T}}x+\lambda^{\mathrm{T}}(b-Ax),$$

则

$$g(x,\lambda)=(c^{\mathrm{T}}-\lambda^{\mathrm{T}}A)x+\lambda^{\mathrm{T}}b$$

为 x 的线性函数. 而 $\lambda^{\mathrm{T}}b$ 为常数，又因它们的约束相同，故 LR 同(6.8)的复杂性相同. 很明显看出 $S\subseteq S_{\mathrm{LR}}$ 且

$$\forall\ \lambda\geqslant 0, x\in S\Rightarrow c^{\mathrm{T}}x+\lambda^{\mathrm{T}}(b-Ax)$$
$$\leqslant c^{\mathrm{T}}x.$$

由定理 6.2 得到 $\forall\ \lambda\geqslant 0\Rightarrow z_{\mathrm{LR}}(\lambda)\leqslant z_{\mathrm{IP}}$. □

定理 6.3 说明拉格朗日松弛是 IP 的下界，但差别有多大？我们的目的是求与 z_{IP} 最接近的下界. 于是需要求解

$$\text{(LD)}\qquad z_{\mathrm{LD}}=\max_{\lambda\geqslant 0} z_{\mathrm{LR}}(\lambda).$$

问题 LD 称为 IP 的拉格朗日对偶. 用下例来理解拉格朗日松弛和对偶等概念. 先定义凸集和凸包的概念.

定义 6.2　若$\forall\ \boldsymbol{x},\boldsymbol{y}\in D$，满足

$$\alpha\boldsymbol{x}+(1-\alpha)\boldsymbol{y}\in D,0\leqslant\alpha\leqslant 1,$$

则称集合 D 为一个凸集.

对离散点集 $Q=\{P_i\mid i=1,2,\cdots\}$，它的凸包定义为

$$\operatorname{con}(Q)=\left\{P=\sum_i\alpha_iP_i\mid\alpha_i\geqslant 0\text{ 实数},\sum_i\alpha_i=1\right\}.$$

很容易验证 con(Q)为凸集.

例 6.2　假设整数规划问题 IP

$$\begin{cases}z_{\mathrm{IP}}=\min -7x_1-2x_2\\ \text{s.t.}\quad x_1-2x_2\geqslant -4\\ \qquad -5x_1-x_2\geqslant -20\\ \qquad 2x_1+2x_2\geqslant 7\\ \qquad x_1\geqslant 2\\ \qquad -x_2\geqslant -4\\ \qquad x\in\mathbf{Z}_+^2.\end{cases}\tag{6.9}$$

的第一个约束为复杂约束，那么拉格朗日松弛后的模型 LR 为

$$\begin{cases}z_{\mathrm{LR}}(\lambda)=\min[-(7+\lambda)x_1-(2-2\lambda)x_2]-4\lambda\\ \text{s.t.}\quad -5x_1-x_2\geqslant -20\quad (l1)\\ \qquad 2x_1+2x_2\geqslant 7\quad (l2)\\ \qquad x_1\geqslant 2\quad (l3)\\ \qquad -x_2\geqslant -4\quad (l4)\\ \qquad x\in\mathbf{Z}_+^2.\end{cases}\tag{6.10}$$

问题(6.10)可以用图解的方法简单求解.图解形式如示意图 6.1.

其中，⊙表示整数点，$ABCDE$ 分别表示(6.10)的可行解集的极点，图中的四条直线分别代表(6.10)四个约束取等号时的直线方程.

当 $\lambda=0$ 时，目标函数的下降方向是(7,2)，图解(6.10)的最优解为(3,4)，$z_{\mathrm{LR}}(0)=-29$.

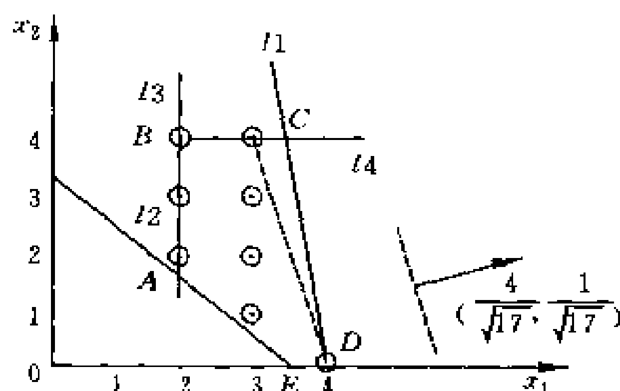


图 6.1 (6.10)的图解示意

当 $\lambda=\frac{1}{2}$ 时，目标函数的下降方向是(7.5，1)，图解(6.10)的最优解为(4，0)，$z_{LR}\left(\frac{1}{2}\right)=-32$.

当 $\lambda=1$ 时，目标函数的下降方向是(8，0)，图解(6.10)的最优解为(4，0)，$z_{LR}(1)=-32$.

由目标函数可知，$(7+\lambda,2-2\lambda)$为目标函数的下降方向. 当 λ 在 $[0,+\infty)$ 变动时，单位化的该方向从 $\left(\frac{7}{\sqrt{53}},\frac{2}{\sqrt{53}}\right)$ 顺时针变化到

$$\lim_{\lambda\to\infty}\left(\frac{7+\lambda}{\sqrt{53+6\lambda+5\lambda^2}},\frac{2-2\lambda}{\sqrt{53+6\lambda+5\lambda^2}}\right)=\left(\frac{1}{\sqrt{5}},\frac{-2}{\sqrt{5}}\right).$$

于是，只可能在两点(3，4)和(4，0)达到最优解. 根据图 6.1 中方向变化求得目标值. 过(3，4)和(4，0)两点的直线方程为 $x_2+4x_1=16$，在图 6.1 中用虚线表示，它的一个垂直方向是 $\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right)$. 此时求得满足

$$\left(\frac{7+\lambda}{\sqrt{53+6\lambda+5\lambda^2}},\frac{2-2\lambda}{\sqrt{53+6\lambda+5\lambda^2}}\right)=\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right)$$

的 $\lambda^* = \frac{1}{9}$. 当 $0 \leqslant \lambda \leqslant \lambda^*$ 时,最优解为(3,4). 当 $\lambda^* \leqslant \lambda$ 时,最优解为(4,0). 综合得到

$$z_{LR}(\lambda) = \begin{cases} -29 + \lambda, & \text{当 } 0 \leqslant \lambda \leqslant \frac{1}{9}, \\ -28 - 8\lambda, & \text{当 } \lambda \geqslant \frac{1}{9}. \end{cases} \tag{6.11}$$

由(6.11)解出(6.9)的拉格朗日对偶问题目标值为

$$z_{LD} = z_{LR}\left(\frac{1}{9}\right) = -28\frac{8}{9}.$$

记(6.9)的简单约束(即(6.10)的约束)的可行解集合为 Q,则 $Q = \{(2,2),(2,3),(2,4)(3,1),(3,2),(3,3),(3,4),(4,0)\}$.

达到 $z_{LD} = -28\frac{8}{9}$ 的极点是 $\lambda = \frac{1}{9}, x = (4,0)$ 和 $\lambda = \frac{1}{9}, x = (3,4)$. 同时,由图 6.1 直观看出,在(4,0)和(3,4)点连接直线上(图 6.1 中用虚线表示)的任何一点都有相同的目标值 $z_{LD} = -28\frac{8}{9}$.

考虑复杂约束 $x_1 - 2x_2 \geqslant -4$,很容易验证:直线 $x_1 - 2x_2 = -4$同过(4,0)和(3,4)两点的直线相交于 $\boldsymbol{x}^* = \left(\frac{28}{9}, \frac{32}{9}\right)$.

综合上面的讨论,$z_{LD} = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*$ 且 $\boldsymbol{x}^* \in \{\boldsymbol{x} \in \mathrm{con}(Q) \mid x_1 - 2x_2 \geqslant -4\}$. □

下面定理说明例 6.2 最后一个结论有普遍性.

定理 6.4 若拉格朗日对偶的目标值 z_{LD}有限,则

$$z_{LD} = \min\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \boldsymbol{x} \in \mathrm{con}(Q)\},$$

其中

$$Q = \{\boldsymbol{x} \mid \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d}, \boldsymbol{x} \in \mathbf{Z}_+^n\}.$$

证明

$$z_{LR}(\boldsymbol{\lambda}) = \min_{x \in Q}(\boldsymbol{c}^{\mathrm{T}} - \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{A})\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{b}$$

$$= \min_{x \in \text{con}(Q)} (c^{\mathrm{T}} - \lambda^{\mathrm{T}} A)x + \lambda^{\mathrm{T}} b$$

$$= \min_{x \in \text{con}(Q)} [c^{\mathrm{T}} x + \lambda^{\mathrm{T}} (b - Ax)].$$

设 con(Q)的极点为$\{x^k | k \in K\}$，极方向为$\{r^j | j \in J\}$，则

$$\min_{x \in Q} (c^{\mathrm{T}} - \lambda^{\mathrm{T}} A)x + \lambda^{\mathrm{T}} b$$

$$= \begin{cases} -\infty, & \text{若存在 } j \in J \text{ 满足} (c^{\mathrm{T}} - \lambda^{\mathrm{T}} A) r^j < 0, \\ c^{\mathrm{T}} x^k + \lambda^{\mathrm{T}} (b - Ax^k), & \text{其它}, k \in K. \end{cases}$$

而由 z_{LD}有限，则有

$$\begin{cases} \text{存在 } \lambda \geqslant 0, \forall\, j \in J, \text{使得} (c^{\mathrm{T}} - \lambda^{\mathrm{T}} A) r^j \geqslant 0, \\ z_{\mathrm{LD}} = \max\limits_{\lambda \geqslant 0} z_{\mathrm{LR}}(\lambda) = \max\limits_{\lambda \geqslant 0} \min\limits_{k \in K} [c^{\mathrm{T}} x^k + \lambda^{\mathrm{T}} (b - Ax^k)]. \end{cases}$$

这一结论等价为

$$\begin{aligned} z_{\mathrm{LD}} = {} & \max \eta \\ \text{s.t.} \quad & c^{\mathrm{T}} x^k + \lambda^{\mathrm{T}} (b - Ax^k) \geqslant \eta, \quad \forall\, k \in K, \\ & (c^{\mathrm{T}} - \lambda^{\mathrm{T}} A) r^j \geqslant 0, \quad \forall\, j \in J, \\ & \lambda \geqslant 0. \end{aligned}$$

整理得到

$$\begin{aligned} z_{\mathrm{LD}} = {} & \max \eta \\ \text{s.t.} \quad & \eta + \lambda^{\mathrm{T}} (Ax^k - b) \leqslant c^{\mathrm{T}} x^k, \quad \forall\, k \in K, \\ & \lambda^{\mathrm{T}} A r^j \leqslant c^{\mathrm{T}} r^j, \quad \forall\, j \in J, \\ & \lambda \geqslant 0. \end{aligned}$$

由线性规划的对偶理论，上式的对偶线性规划为

$$\begin{aligned} z_{\mathrm{LD}} = {} & \min c^{\mathrm{T}} \Big(\sum_{k \in K} \alpha_k x^k + \sum_{j \in J} \beta_j r^j \Big) \\ \text{s.t.} \quad & \sum_{k \in K} \alpha_k = 1, \\ & A \Big(\sum_{k \in K} \alpha_k x^k + \sum_{j \in J} \beta_j r^j \Big) \geqslant b \Big(\sum_{k \in K} \alpha_k \Big), \\ & \alpha_k \geqslant 0, k \in K; \beta_j \geqslant 0, j \in J. \end{aligned}$$

最终整理得到

$$z_{\mathrm{LD}} = \min_{x \in \mathrm{con}(Q)} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$$

$$\text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b} \qquad \square$$

$$= \min\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \boldsymbol{x} \in \mathrm{con}(Q)\}.$$

推论 6.1 对任给 $\boldsymbol{c}$，整数规划问题 IP 和拉格朗日对偶 LD 的目标值 $z_{\mathrm{IP}} = z_{\mathrm{LD}}$ 成立的充要条件是

$$\mathrm{con}(Q \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}) = \mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}.$$

证明 很容易看出

$$Q \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\} \subseteq \mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\},$$

得到

$$\begin{aligned} &\mathrm{con}(Q \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}) \\ &\subseteq \mathrm{con}(\mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}) \\ &= \mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}. \end{aligned}$$

再由定理 6.4，得到

$$z_{\mathrm{IP}} = \min_{x \in \mathrm{con}(Q \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\})} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leqslant z_{\mathrm{LD}} = \min_{x \in \mathrm{con}(Q) \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\}} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}.$$

若对任给的 $\boldsymbol{c}$ 满足：$z_{\mathrm{IP}} = z_{\mathrm{LD}}$，则得到

$$\mathrm{con}(Q \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}) = \mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}\}.$$

由此证明了必要性．由上面的推导，充分性明显成立． □

例 6.3 （续例 6.2）例 6.2 中的

$$Q = \{(2,2),(2,3),(2,4),(3,1),(3,2),(3,3),(3,4),(4,0)\}.$$

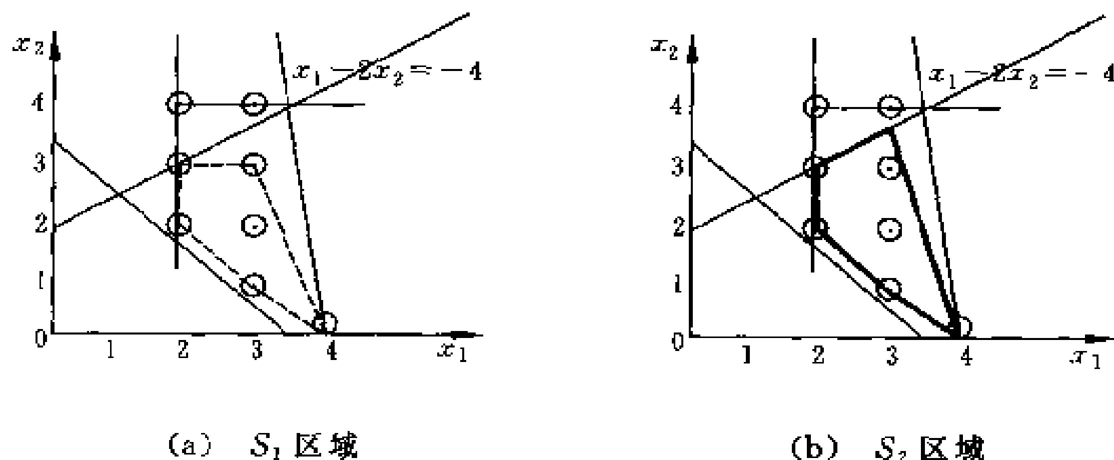
记

$$S_1 = \mathrm{con}(Q \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid x_1 - 2x_2 \geqslant -4\}),$$

$$S_2 = \mathrm{con}(Q) \cap \{\boldsymbol{x} \in \mathbf{R}_+^n \mid x_1 - 2x_2 \geqslant -4\}.$$

它们的区域图形表示见图 6.2．S_1 区域为图 6.2(a)的虚线所围部分．S_2 区域为图 6.2(b)的黑粗线所围部分．

由推论 6.1 可以看出，$z_{\mathrm{IP}} - z_{\mathrm{LD}}$ 的差值同下面两个因素有关．第一是目标函数中的 $\boldsymbol{c}$．推论 6.1 中是对所有的 $\boldsymbol{c}$ 讨论，但可能存



(a)　S_1 区域　　(b)　S_2 区域

图　6.2

在一个 $\boldsymbol{c}$ 使得 $z_{\mathrm{IP}}=z_{\mathrm{LD}}$，如在例 6.2 中 $\boldsymbol{c}^{\mathrm{T}}=(1,0)$时，则，

$$
\begin{aligned}
z_{\mathrm{IP}} = & \min x_1 \\
\text{s.t.}\quad & x_1 - 2x_2 \geqslant -4, \\
& -5x_1 - x_2 \geqslant -20, \\
& 2x_1 + 2x_2 \geqslant 7, \\
& x_1 \geqslant 2, \\
& -x_2 \geqslant -4, \\
& x \in \mathbf{Z}_+^2.
\end{aligned}
$$

$$
\begin{aligned}
z_{\mathrm{LR}}(\lambda) = & \min[(1-\lambda)x_1 + 2\lambda x_2] - 4\lambda \\
\text{s.t.}\quad & -5x_1 - x_2 \geqslant -20, \\
& 2x_1 + 2x_2 \geqslant 7, \\
& x_1 \geqslant 2, \\
& -x_2 \geqslant -4, \\
& x \in \mathbf{Z}_+^2.
\end{aligned}
$$

解出 $z_{\mathrm{IP}}=2$. (2,2)和(2,3)两点的直线方程为 $x=2$，它的一个垂直方向是(−1,0). 此时，求满足

$$\left(\frac{\lambda-1}{\sqrt{1-2\lambda+5\lambda^2}},\frac{-2\lambda}{\sqrt{1-2\lambda+5\lambda^2}}\right)=(-1,0)$$

的 $\lambda^*=0$. 当 $\lambda=\lambda^*=0$ 时，松驰的一个最优解为(2,2). 于是 $z_{LR}(\lambda^*)=2$.

第二个因素是可行解的区域. 根据图 6.2(a)和图 6.2(b)中的 S_1 区域和 S_2 区域不同，存在 $\boldsymbol{c}$ 使得 $z_{IP}-z_{LD}$不为零. 如例 6.2，$z_{LD}=-28\frac{8}{9}$，在 $\lambda^*=\frac{1}{9}$达到拉格朗日对偶问题的最优值，它的一个最优解是(4,0)；$z_{IP}=-28$，一个最优解也为(4,0). 从这个结论可以看出，即使拉格朗日松弛在某一个 λ 求到的最优解为原问题(6.9)的可行解，我们不能断言 $z_{IP}=z_{LR}(\lambda)$. □

如果拉格朗日对偶在 $\lambda=0$ 时达到最优值，且它的最优解是 IP 的一个可行解，此时，原问题 IP 同松弛问题 LR 为同一个目标函数，所以有 $z_{IP}=z_{LR}(\lambda)$.

IP 的线性规划松弛为

$$\begin{aligned}(\text{LP})\qquad & z_{LP}=\min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\\ & \text{s.t. } \boldsymbol{A}\boldsymbol{x}\geqslant \boldsymbol{b}(\text{复杂约束}),\\ & \boldsymbol{B}\boldsymbol{x}\geqslant \boldsymbol{d}(\text{简单约束}),\\ & \boldsymbol{x}\in \mathbf{R}_+^n.\end{aligned}$$

继续研究例 6.2 还可以发现：已经得到 $z_{IP}=-28$，$z_{LD}=-28\frac{8}{9}$. 再用图解法，可得到线性规划松弛的最优解为$\left(\frac{36}{11},\frac{40}{11}\right)$，最优值为 $z_{LP}=-30\frac{2}{11}$.

定理 6.5 若线性规划松弛 LP 存在可行解，则 $z_{LP}\leqslant z_{LD}\leqslant z_{IP}$.

证明 因为

$$\begin{aligned}&Q\cap\{\boldsymbol{x}\in \mathbf{R}_+^n\mid \boldsymbol{A}\boldsymbol{x}\geqslant \boldsymbol{b}\}\\ &\quad\subseteq \operatorname{con}(Q)\cap\{\boldsymbol{x}\in \mathbf{R}_+^n\mid \boldsymbol{A}\boldsymbol{x}\geqslant \boldsymbol{b}\}\end{aligned}$$

$$\subseteq \{x \in \mathbf{R}_+^n \mid Bx \geqslant d\}$$
$$\cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\},$$

得到

$$\text{con}(Q \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\}),$$
$$\subseteq \text{con}(Q) \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\},$$
$$\subseteq \{x \in \mathbf{R}_+^n \mid Bx \geqslant d\}$$
$$\cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\},$$

再由定理 6.4，推出

$$z_{\text{IP}} \geqslant z_{\text{LD}} \geqslant z_{\text{LP}}.$$ □

由定理 6.5 得到，采用拉格朗日松弛对偶后的目标值 z_{LD} 是 IP 的一个下界且不比 z_{LP} 差，因此可能出现差距 $z_{\text{LD}}-z_{\text{LP}}$. 这解释了例 6.3 结束时给出的直观结论.

推论 6.2　若 $\{x\in\mathbf{R}_+^n \mid Bx\geqslant d\}$ 的所有极点为整数点，则对任意的 c 有 $z_{\text{LD}}=z_{\text{LP}}$.

证明　从定理 6.5 的证明过程中得到

$$\text{con}(Q) \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\} \subseteq \{x \in \mathbf{R}_+^n \mid Bx \geqslant d\}$$
$$\cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\},$$

当 $\{x\in\mathbf{R}_+^n \mid Bx\geqslant d\}$ 的所有极点为整数点时，

$$\{x \in \mathbf{R}_+^n \mid Bx \geqslant d\} = \text{con}(Q),$$

于是

$$\text{con}(Q) \cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\} = \{x \in \mathbf{R}_+^n \mid Bx \geqslant d\}$$
$$\cap \{x \in \mathbf{R}_+^n \mid Ax \geqslant b\},$$

再由定理 6.4 得到结论. □

定理 6.5 告诉我们，拉格朗日对偶的目标值是 IP 问题的一个下界，那么，这个下界与 IP 的目标值有多大的差距？

定理 6.6 $z_{\mathrm{IP}}-z_{\mathrm{LD}}\leqslant\varepsilon(\varepsilon\geqslant 0)$的充分必要条件是存在

$$\boldsymbol{\lambda}^{*}\geqslant 0 \text{ 和 } \boldsymbol{x}^{*}\in\{\boldsymbol{x}\in \boldsymbol{Z}_{+}^{n}\mid \boldsymbol{A}\boldsymbol{x}\geqslant \boldsymbol{b},\boldsymbol{B}\boldsymbol{x}\geqslant \boldsymbol{d}\}$$

满足

$$\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})\geqslant -\delta_1(\delta_1\geqslant 0),$$
$$z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{*}+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})\leqslant z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})+\delta_2(\delta_2\geqslant 0),$$
$$\delta_1+\delta_2\leqslant\varepsilon.$$

证明 充分性:

$$\begin{aligned}z_{\mathrm{LD}}&\geqslant z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})\geqslant z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})-\delta_2\\&\geqslant \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{*}-\delta_1-\delta_2\geqslant z_{\mathrm{IP}}-\delta_1-\delta_2.\end{aligned}$$

必要性:

记 $\boldsymbol{x}^{*}$ 为 IP 的最优解,$\boldsymbol{\lambda}^{*}$ 为 LD 的最优解,则有

$$\begin{aligned}z_{\mathrm{LD}}&=z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{*}+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})+z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})-z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})\\&=z_{\mathrm{IP}}+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})+z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})-z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*}).\end{aligned}$$

由条件 $z_{\mathrm{IP}}-z_{\mathrm{LD}}\leqslant\varepsilon(\varepsilon\geqslant 0)$推出

$$\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})+z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})-z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})\geqslant -\varepsilon,$$

记

$$\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})=-\delta_1,z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})-z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})=-\delta_2,$$

则得到

$$\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})=-\delta_1(\delta_1\geqslant 0),$$
$$z(\boldsymbol{\lambda}^{*},\boldsymbol{x}^{*})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^{*}+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})=z_{\mathrm{LR}}(\boldsymbol{\lambda}^{*})+\delta_2(\delta_2\geqslant 0),$$
$$\delta_1+\delta_2\leqslant\varepsilon.$$

□

可以用定理 6.6 的充分条件来估计一个算法同拉格朗日松弛下界的距离.

例 6.4 (续例 6.2) 当 $\lambda^{*}=\frac{1}{9}$时,$\boldsymbol{x}^{*}=(4,0)^{\mathrm{T}}$ 为(6.9)问题的一个可行解,此时,

$$\lambda^{*}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{*})=\frac{1}{9}(-4-4)=-\delta_1,$$

$$z(\lambda^*,x^*)=c^Tx^*+\lambda^*(b-Ax^*)$$
$$=-28-\frac{8}{9}=-28\frac{8}{9}=z_{LR}(\lambda^*)+\delta_2,$$

其中 $\delta_2=0$. 于是 $\varepsilon=\delta_1+\delta_2=\frac{8}{9}$,故有 $z_{IP}-z_{LD}\leqslant\frac{8}{9}$.

一般的情况下,无法估计得那么精确,但也可以用定理 6.6 的充分条件估计 $z_{IP}-z_{LD}$. 如 $\lambda^*=\frac{1}{2}$时,$x^*=(4,0)^T$ 为(6.9)问题的一个可行解,此时,

$$\lambda^*(b-Ax^*)=\frac{1}{2}(-4-4)=-\delta_1,$$
$$z(\lambda^*,x^*)=c^Tx^*+\lambda^*(b-Ax^*)$$
$$=-28-4=-32=z_{LR}(\lambda^*)+\delta_2,$$

其中,$\delta_2=-32-z_{LR}(\lambda^*)=-32+28+4=0$. 于是 $\varepsilon=\delta_1+\delta_2=4$,故有 $z_{IP}-z_{LD}\leqslant 4$. □

从上面的讨论看出,只有给出较好的 λ^* 和 x^*,才能得到 $z_{IP}-z_{LD}$较好的估计值. 在有多种约束组合可松弛的拉格朗日松弛中,我们的目的是选 $z_{IP}-z_{LD}$值最小的松驰.

6.3 拉格朗日松弛的进一步讨论

在前二节中,我们仅就标准的拉格朗日松弛进行了讨论. 在实际应用中还可能出现其他的形式. 本节针对一些常见的问题分类讨论.

1. 等号约束的松弛

将等号约束

$$\sum_{j=1}^{n}a_{ij}x_j=b_i,$$

写成两个标准形式

$$\sum_{j=1}^{n} a_{ij}x_j \geqslant b_i \text{和} \sum_{j=1}^{n}(-a_{ij})x_j \geqslant -b_i.$$

对应的拉格朗日乘子 $\lambda_{i1} \geqslant 0, \lambda_{i2} \geqslant 0$，及在目标函数中出现

$$\lambda_{i1}\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right) + \lambda_{i2}\left(-b_i - \sum_{j=1}^{n}(-a_{ij})x_j\right)$$

$$= (\lambda_{i1} - \lambda_{i2})\left(b_i - \sum_{j=1}^{n} a_{ij}x_j\right).$$

令 $\lambda_i = \lambda_{i1} - \lambda_{i2}$，则松弛后对应的 λ_i 没有正负号约束.

2. LR 最优解同 IP 最优解的关系

若对给定的 $\boldsymbol{\lambda} \geqslant 0$，

$$\text{(LR)} \quad \begin{aligned} z_{\text{LR}}(\boldsymbol{\lambda}) = &\min \boldsymbol{c}^{\text{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\text{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \\ &\text{s.t.}\ \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d} \\ &\boldsymbol{x} \in \mathbf{Z}_+^n. \end{aligned}$$

的最优解 $\boldsymbol{x}(\boldsymbol{\lambda})$ 为 IP 的一个可行解，则有 $z_{\text{LR}}(\boldsymbol{\lambda}) \leqslant z_{\text{IP}}$. 值得注意的是并不能得到 $\boldsymbol{c}^{\text{T}}\boldsymbol{x}(\boldsymbol{\lambda}) = z_{\text{IP}}$，用下例说明这个结论.

例 6.5 集合覆盖问题

$$\begin{cases} \min 2x_1 + 3x_2 + 4x_3 + 5x_4 \\ \text{s.t.}\ x_1 + x_3 \geqslant 1, \\ \quad x_1 + x_4 \geqslant 1, \\ \quad x_2 + x_3 + x_4 \geqslant 1, \\ \quad x_i \in \{0,1\},\ i = 1,2,3,4, \end{cases} \tag{6.12}$$

直观的结果是最优解为 $x_1 = x_2 = 1, x_3 = x_4 = 0, z_{\text{IP}} = 5$. 拉格朗日松弛三个约束.

$$\begin{cases} z_{\text{LR}}(\boldsymbol{\lambda}) = \min(2 - \lambda_1 - \lambda_2)x_1 + (3 - \lambda_3)x_2 \\ \qquad + (4 - \lambda_1 - \lambda_3)x_3 + (5 - \lambda_2 - \lambda_3)x_4 + \lambda_1 + \lambda_2 + \lambda_3 \\ \text{s.t.}\ x_j \in \{0,1\},\ j = 1,2,3,4. \end{cases} \tag{6.12'}$$

当 $\lambda_1=\lambda_2=\lambda_3=4$ 时，

$$z_{LR}(\lambda)=\min -6x_1-x_2-4x_3-3x_4+12$$
$$\text{s.t. } x_j\in\{0,1\}, j=1,2,3,4,$$

其最优解为：$x_1=x_2=x_3=x_4=1, z_{IP}=-2$. 该解为(6.12)问题的一个可行解，但

$$c^{T}x=(1,2,3,4)\begin{bmatrix}1\\1\\1\\1\end{bmatrix}=10\neq z_{IP}.$$

由此得到：当松弛后问题的一个最优解为原问题(6.12)的一个可行解时，并不能得到该解为原问题的最优解. □

当拉格朗日松弛后的一个最优解为原问题 IP 的一个可行解时，在什么样的条件下，该解为 IP 的一个最优解？

推论 6.3　$z_{IP}=z_{LD}$的充分必要条件是存在 $\boldsymbol{\lambda}^*\geqslant 0$ 和 $\boldsymbol{x}^*$ 为 IP 的可行解，满足$\boldsymbol{\lambda}^{*T}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^*)=0$ 且 $z_{LR}(\boldsymbol{\lambda}^*)=z(\boldsymbol{\lambda}^*,\boldsymbol{x}^*)$.

证明　定理 6.6 的直接推论. □

当拉格朗日松弛后对应 $\boldsymbol{\lambda}^*\geqslant 0$ 的一个最优解 $\boldsymbol{x}^*$ 为原问题 IP 的一个可行解时，此时已有 $z_{LR}(\boldsymbol{\lambda}^*)=z(\boldsymbol{\lambda}^*,\boldsymbol{x}^*)$，只需 $\boldsymbol{\lambda}^{*T}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^*)=0$ 时，有 $z_{IP}=z_{LD}$. 这是对上面问题的回答.

3. 拉格朗日松弛的整数性

定义 6.3　若 LR($\forall\ \lambda\geqslant 0$)的最优解与其整数约束 $\boldsymbol{x}\in\boldsymbol{Z}_+^n$ 无关，即

$$z_{LR}(\boldsymbol{\lambda})=\min \boldsymbol{c}^{T}\boldsymbol{x}+\boldsymbol{\lambda}^{T}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})$$
$$\text{s.t. } \boldsymbol{B}\boldsymbol{x}\geqslant \boldsymbol{d},$$
$$\boldsymbol{x}\in\boldsymbol{Z}_+^n,$$

与 LR 的线性松弛

$$(\text{LRL})\begin{cases} z_{\text{LRL}}(\boldsymbol{\lambda}) = \min \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}) \\ \quad \text{s.t.}\ \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d}, \\ \quad \boldsymbol{x} \in \mathbf{R}_{+}^{n}, \end{cases}$$

的最优值相同，则称该问题的拉格朗日松弛具有整数性.

从整数性可以看出，若 LR 具有整数性，则对任意的 $\boldsymbol{\lambda} \geqslant 0$，最优解在 $\{\boldsymbol{x} \in \mathbf{R}_{+}^{n} \mid \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d}\}$ 的一个极点达到且该极点一定是整数点.

例 6.6 （续 6.1）例 6.1 的集合覆盖问题 SC 的拉格朗日松弛为

$$(\text{LRSC})\begin{cases} z_{\text{LRSC}}(\boldsymbol{\lambda}) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ \text{s.t.}\quad x_j \in \{0,1\}, j = 1,2,\cdots,n, \end{cases}$$

其中

$$d_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij}.$$

上面公式的线性规划模型为

$$(\text{LSC})\begin{cases} z_{\text{LSC}}(\boldsymbol{\lambda}) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ \text{s.t.}\quad 0 \leqslant x_j \leqslant 1, j = 1,2,\cdots,n. \end{cases}$$

LRSC 和 LSC 的最优解都为

$$x_j^* = \begin{cases} 1, & \text{若 } d_j \leqslant 0, \\ 0, & \text{其它}. \end{cases}$$

由此可知 LRSC 具有整数性. □

推论 6.4 若 LR 具有整数性，则 $z_{\text{LD}} = z_{\text{LP}}$.

证明 记

$$Q' = \{\boldsymbol{x} \in \mathbf{R}_{+}^{n} \mid \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d}\},$$

$$z_{\text{LRL}}(\boldsymbol{\lambda}) = \min_{x \in Q'} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}),$$

$$z'_{\text{LD}} = \max_{\lambda \geqslant 0} z_{\text{LRL}}(\boldsymbol{\lambda}).$$

同定理 6.4 的证明相同，有

$$z'_{LD}=\min\{c^T x \mid Ax\geqslant b, x\in \operatorname{con}(Q')\}$$
$$=\min\{c^T x \mid Ax\geqslant b, x\in Q'\}=z_{LP}.$$

由 $z_{LRL}(\lambda)$定义，存在 $\lambda^*\geqslant 0$ 使

$$z_{LP}=z'_{LD}=z_{LRL}(\lambda^*).$$

由整数性得到：

$$z_{LP}=z_{LRL}(\lambda^*)=z_{LR}(\lambda^*)\leqslant z_{LD}.$$

再由定理 6.5 得到 $z_{LP}=z_{LD}$. □

推论 6.4 同推论 6.2 的结论极为相似，值得注意的是推论 6.2 要求 $Q'=\{x\in \mathbf{R}_+^n \mid Bx\geqslant d\}$的所有极点为整数点，这样可以保证 $z_{LP}=z_{LD}$，说明拉格朗日对偶问题的最优值与 IP 问题的线性规划松弛最优值无差距. 但推论 6.4 的条件比推论 6.2 更弱一些，并不要求 $Q'=\{x\in \mathbf{R}_+^n \mid Bx\geqslant d\}$的所有极点为整数点，只需对应任一 λ，LRL 的最优解为整数点，达到最优目标值 $z_{LRL}(\lambda)$. 这一现象可以从例 6.7 中看出. 由此得到的结论是 $z_{LP}=z_{LD}$，表示线性规划松弛的最优值同拉格朗日对偶的最优值相同，即采用拉格朗日松弛的效果不比线性规划松弛好.

例 6.7 整数规划问题和它的拉格朗日松弛分别为(6.13)和(6.14).

$$\left\{\begin{array}{ll} z_{IP}=\min -7x_1-2x_2 & \\ \text{s.t.}\quad x_1-2x_2\geqslant -4, & \\ -4x_1-x_2\geqslant -16, & \\ 2x_1+2x_2\geqslant 7, & \\ x_1\geqslant 2, & \\ -x_2\geqslant -4, & \\ x\in \mathbf{Z}_+^2. & \end{array}\right. \tag{6.13}$$

$$
\begin{cases}
z_{LR}(\lambda) = \min[-(7+\lambda)x_1 - (2-2\lambda)x_2] - 4\lambda \\
\quad \text{s.t.} \quad -4x_1 - x_2 \geqslant -16, \\
\quad\quad 2x_1 + 2x_2 \geqslant 7, \\
\quad\quad x_1 \geqslant 2, \\
\quad\quad -x_2 \geqslant -4, \\
\quad\quad x \in \mathbf{Z}_+^2.
\end{cases}
\tag{6.14}
$$

本例同例 6.2 非常相似，只是将例 6.2 中问题(6.9)的第二个约束 $-5x_1-x_2\geqslant-20$ 变更为 $-4x_1-x_2\geqslant-16$，按推论 6.4 的定义，

$$
\begin{aligned}
Q' &= \{x\in \mathbf{R}_+^n \mid Bx\geqslant d\} \\
&= \{x\in \mathbf{R}_+^n \mid -4x_1-x_2\geqslant-16, 2x_1+2x_2\geqslant7, x_1\geqslant2, -x_2\geqslant-4\}.
\end{aligned}
$$

用图 6.3 表示解集合 Q' 包括：粗黑线内部和边界.

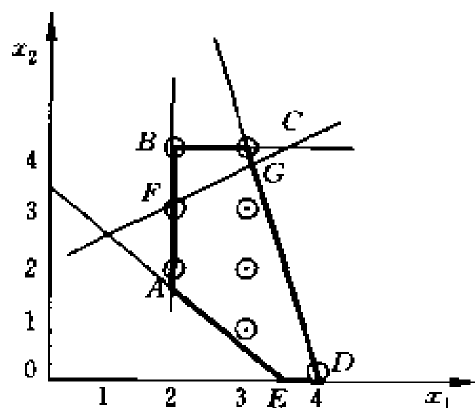


图 6.3 Q' 的解区域示意图

很明显，它的极点为 A、B、C、D、E，且 $A=(2,3/2)$ 和 $E=(7/2,0)$ 不是整数点. 因此，由推论 6.2 无法得到 $z_{LP}=z_{LD}$.

与例 6.2 相同的分析，得到 $\lambda^*=\dfrac{1}{9}$. 当 $0\leqslant\lambda\leqslant\lambda^*$ 时，最优解为 $(3,4)$. 当 $\lambda^*\leqslant\lambda$ 时，最优解为 $(4,0)$. 而 LRL 问题在本例的可行解区域为图 6.3 的粗黑线内部，同上面相同的讨论，他们对每一个 $\lambda\geqslant0$ 的最优值相同，即具有整数性. 于是

$$z_{LR}(\lambda)=z_{1RL}(\lambda)=\begin{cases}-29+\lambda, & \text{当 } 0\leqslant\lambda\leqslant\dfrac{1}{9},\\ -28-8\lambda, & \text{当 } \lambda\geqslant\dfrac{1}{9},\end{cases}$$

解得(6.13)的拉格朗日对偶问题目标值为

$$z_{LD}=z_{LR}\left(\frac{1}{9}\right)=-28\,\frac{8}{9}.$$

线性规划松弛的可行解区域为

$$S'=\left\{\ \boldsymbol{x}\in\mathbf{R}_{+}^{2}\ \middle|\ \begin{array}{l}x_1-2x_2\geqslant-4,\ -4x_1-x_2\geqslant-16,\\ 2x_1+2x_2\geqslant7,x_1\geqslant2,\ -x_2\geqslant-4.\end{array}\right\}$$

它的所有极点为图 6.3 中的 A、F、G、D 和 E，即极点是

$$\left\{\left(2,\frac{3}{2}\right),(2,3),\left(\frac{28}{9},\frac{32}{9}\right),(4,0),\left(\frac{7}{2},0\right)\right\},$$

于是最优值 $z_{LP}=-7\times\dfrac{28}{9}-2\times\dfrac{32}{9}=-28\,\dfrac{8}{9}=z_{LD}$. 验证了推论 6.4 的结论. □

在例 6.7 的条件下，

$$z_{LD}=z_{LP}=-28\,\frac{8}{9}\neq z_{IP}=-28,$$

但并不是说 $Q'=\{\boldsymbol{x}\in\mathbf{R}_{+}^{n}\mid \boldsymbol{Bx}\geqslant\boldsymbol{d}\}$ 的某一个极点不是整数点时，不存在 $z_{LD}=z_{LP}=z_{IP}$.

例 6.8 （续例 6.7）继续例 6.7 的讨论，将(6.13)中的约束 $x_1-2x_2\geqslant-4$ 更改为 $x_1-x_2\geqslant-1$，即模型为：

$$\begin{cases} z_{IP}=\min\ -7x_1-2x_2\\ \quad \text{s.t.}\quad x_1-x_2\geqslant-1,\\ \qquad -4x_1-x_2\geqslant-16,\\ \qquad 2x_1+2x_2\geqslant7,\\ \qquad x_1\geqslant2,\\ \qquad -x_2\geqslant-4,\\ \qquad x\in\mathbf{Z}_{+}^{2}.\end{cases}\tag{6.15}$$

$$\begin{cases} z_{\mathrm{LR}}(\lambda) = \min[-(7+\lambda)x_1 - (2-\lambda)x_2] - \lambda \\ \quad \text{s.t.} \quad -4x_1 - x_2 \geqslant -16, \\ \quad\quad 2x_1 + 2x_2 \geqslant 7, \\ \quad\quad x_1 \geqslant 2, \\ \quad\quad -x_2 \geqslant -4, \\ \quad\quad x \in \mathbf{Z}_+^2. \end{cases} \tag{6.16}$$

图 6.3 中的 A 点 $\left(2,\frac{3}{2}\right)$ 是 $Q'=\{x\in\mathbf{R}_+^n \mid Bx\geqslant d\}$ 的一个极点. 与例 6.2 和例 6.7 相同的讨论,拉格朗日松弛问题(6.16)目标函数下降的方向从 $\left(\frac{7}{\sqrt{53}},\frac{2}{\sqrt{53}}\right)$ 顺时针变化到

$$\lim_{\lambda\to\infty}\left(\frac{7+\lambda}{\sqrt{53+10\lambda+2\lambda^2}},\frac{2-\lambda}{\sqrt{53+10\lambda+2\lambda^2}}\right)=\left(\frac{1}{\sqrt{2}},\frac{-1}{\sqrt{2}}\right).$$

于是,只可能在(3,4)和(4,0)两点达到最优解. 可以根据图中方向变化求得目标值. (3,4)和(4,0)两点的直线方程为 $y+4x=16$,它的一个垂直方向是 $\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right)$. 此时求解 λ 满足

$$\left(\frac{7+\lambda}{\sqrt{53+10\lambda+2\lambda^2}},\frac{2-\lambda}{\sqrt{53+10\lambda+2\lambda^2}}\right)=\left(\frac{4}{\sqrt{17}},\frac{1}{\sqrt{17}}\right),$$

得到 $\lambda^*=\frac{1}{5}$. 当 $0\leqslant\lambda\leqslant\lambda^*$ 时,最优解为(3,4). 当 $\lambda^*\leqslant\lambda$ 时,最优解为(4,0). 于是

$$z_{\mathrm{LR}}(\lambda)=\begin{cases} -29, & \text{当 } 0\leqslant\lambda\leqslant\frac{1}{5}, \\ -28-5\lambda, & \text{当 } \lambda\geqslant\frac{1}{5}. \end{cases}$$

由此解(6.15),得拉格朗日对偶问题目标值为

$$z_{\mathrm{LD}}=z_{\mathrm{LR}}\left(\frac{1}{5}\right)=-29.$$

很容易验证 $x^*=(3,4)$ 为 IP 和 LP 的最优解,目标值满足 z_{LD}

$= z_{LP} = z_{IP} = -29.$ □

4. 拉格朗日分解

拉格朗日分解(decomposition)的基本思想是:将整数规划问题 IP 改写成

$$
\begin{cases}
z_{IP} = \min \boldsymbol{c}^{T}\boldsymbol{x} \\
\quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \\
\quad \boldsymbol{x} = \boldsymbol{y}, \\
\quad \boldsymbol{B}\boldsymbol{y} \geqslant \boldsymbol{d}, \\
\quad \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{Z}_{+}^{n},
\end{cases}
\tag{6.17}
$$

其中,增加辅助变量 $\boldsymbol{y}$,使得约束 $\boldsymbol{x}=\boldsymbol{y}$ 连接 $\boldsymbol{A}\boldsymbol{x}\geqslant\boldsymbol{b}$ 和 $\boldsymbol{B}\boldsymbol{x}\geqslant\boldsymbol{d}$ 两组约束. 采用拉格朗日松弛的方法去掉(6.17)中的约束 $\boldsymbol{x}=\boldsymbol{y}$,对任意 $\boldsymbol{\lambda}$ 得

$$
\begin{cases}
\min \boldsymbol{c}^{T}\boldsymbol{x} + \boldsymbol{\lambda}^{T}(\boldsymbol{x} - \boldsymbol{y}) \\
\text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \\
\quad \boldsymbol{B}\boldsymbol{y} \geqslant \boldsymbol{d}, \\
\quad \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{Z}_{+}^{n}.
\end{cases}
\tag{6.18}
$$

由目标函数的线性性,(6.18)等价于

$$
\begin{cases}
z_{LR1}(\boldsymbol{\lambda}) = \min \boldsymbol{c}^{T}\boldsymbol{x} + \boldsymbol{\lambda}^{T}\boldsymbol{x} \\
\quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \\
\quad \boldsymbol{x} \in \boldsymbol{Z}_{+}^{n},
\end{cases}
\tag{6.19}
$$

和

$$
\begin{cases}
z_{LR2}(\boldsymbol{\lambda}) = \min -\boldsymbol{\lambda}^{T}\boldsymbol{y} \\
\quad \text{s.t. } \boldsymbol{B}\boldsymbol{y} \geqslant \boldsymbol{d}, \\
\quad \boldsymbol{y} \in \boldsymbol{Z}_{+}^{n}.
\end{cases}
\tag{6.20}
$$

IP 问题松弛并且分解为(6.19)和(6.20)两个整数规划问题. 它们每一个选 IP 的一部分约束作为自己的约束,使得每一个问题的计算规模减少. 更重要的是达到分解的目的:寻求(6.19)和

(6.20)是可以简单求解的整数规划问题.这同拉格朗日松弛的原理相同.(6.19)和(6.20)称为 IP 的拉格朗日分解.

由于拉格朗日分解是拉格朗日松弛的一种特殊情况,很容易验证

$$z_{LR1}(\boldsymbol{\lambda}) + z_{LR2}(\boldsymbol{\lambda}) \leqslant z_{IP}.$$

它的对偶问题是

$$z_{LD} = \max_{\lambda} z_{LR1}(\boldsymbol{\lambda}) + z_{LR2}(\boldsymbol{\lambda}). \tag{6.21}$$

有关的理论和计算同拉格朗日松弛相同.

5. 增加拉格朗日松弛的约束

有时,拉格朗日松弛使得约束过于宽松,此时增加一些约束可以提高松弛后问题的计算效果.值得注意的是不能因为增加约束而使得问题的复杂性有本质的不同,即拉格朗日松弛问题从一个多项式问题转化为 NP-hard 问题.

例 6.9 (续例 6.1) 当集合覆盖问题松弛所有的约束时,拉格朗日松弛问题为

$$\left\{\begin{array}{l} z_{LRSC1}(\boldsymbol{\lambda}) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ \text{s.t.} \quad x_j \in \{0,1\},\ j = 1,2,\cdots,n, \\ \qquad \boldsymbol{\lambda} \geqslant 0. \end{array}\right. \tag{6.22}$$

其中

$$d_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij},$$

最优解为

$$x_j^* = \begin{cases} 1, & \text{若 } d_j \leqslant 0, \\ 0, & \text{其他}. \end{cases}$$

从集合覆盖问题的本身可知:任意选定一列,任意选定一个非零的位置,则非零元素对应的行被选用的列覆盖;由此,n 列中存

在不超过 m 列的一个列集合(每个列只少有一个 1)覆盖 m 行. 增加约束

$$1 \leqslant \sum_{j=1}^{n} x_j \leqslant m,$$

使得(6.22)变为

$$\begin{cases} z_{LRSC2}(\lambda) = \min \sum_{j=1}^{n} d_j x_j + \sum_{i=1}^{m} \lambda_i \\ \text{s.t.} \quad 1 \leqslant \sum_{j=1}^{n} x_j \leqslant m, \\ \qquad x_j \in \{0,1\}, j = 1,2,\cdots,n, \\ \qquad \lambda \geqslant 0. \end{cases} \tag{6.23}$$

不难证明,(6.23)是集合覆盖 SC 的一个松弛,且对任意 $\lambda \geqslant 0$,有

$$z_{LRSC1}(\lambda) \leqslant z_{LRSC2}(\lambda) \leqslant z_{SC}.$$

□

6.4 拉格朗日松弛算法

拉格朗日松弛算法主要包括次梯度优化算法和拉格朗日松弛启发式算法. 它们的两个主要应用是给出 IP 问题的下界和构造基于拉格朗日松弛的启发式算法. 由前面的讨论可知,每一个 λ 对应的 $z_{LR}(\lambda)$都可以作为 IP 的下界,下界的最佳值为 z_{LD}. 求解 z_{LD}的过程采用类似非线性规划的梯度——次梯度优化算法. 进一步可以将次梯度算法扩展为拉格朗日松弛启发式算法. 它的两个主要步骤是:第一步是确定一个 λ 及求解对应 LR 的最优解 x;第二步是当 x 不是 IP 的可行解时,将其可行化.

6.4.1 次梯度优化算法

次梯度优化(subgradient optimization)算法的思想与非线性

规划的梯度下降思想相同. 拉格朗日对偶问题希望 IP 的下界 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 尽可能大，于是按 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 的上升方向渐渐逼近 z_{LD}. 次梯度优化算法就是根据 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 本身的分段线性而构造.

定义 6.4 函数 $g:\mathbf{R}^m\to\mathbf{R}$ 满足

$$\forall\ x_1,x_2\in\mathbf{R}^m,$$

$$0\leqslant\alpha\leqslant 1: g(\alpha x^1+(1-\alpha)x^2)\geqslant \alpha g(x^1)+(1-\alpha)g(x^2),$$

则称 $g(x)$ 为凹函数.

定理 6.7 若 LR 的可行解集合 Q 是有限个整数点集合，函数

$$z_{\mathrm{LR}}(\boldsymbol{\lambda})=\min\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})\mid \boldsymbol{x}\in Q\}$$

是凹函数.

证明 由于 Q 是有限个整数点的集合，记这些点为 $\boldsymbol{x}^k(k=1,2,\cdots,K)$，于是

$$z_{\mathrm{LR}}(\boldsymbol{\lambda})=\min_{1\leqslant k\leqslant K}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^k+\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^k)\}.$$

记达到最优值的点为 $\boldsymbol{x}^q$，则

$$z_{\mathrm{LR}}(\boldsymbol{\lambda})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^q+\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^q).$$

对 $\forall\ \boldsymbol{\lambda}^1,\boldsymbol{\lambda}^2$ 及 $\boldsymbol{\lambda}=\alpha\boldsymbol{\lambda}^1+(1-\alpha)\boldsymbol{\lambda}^2, 0\leqslant\alpha\leqslant 1$，有

$$\begin{aligned}z_{\mathrm{LR}}(\boldsymbol{\lambda}) &= \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^q+\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^q)\\ &=\alpha[\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+(\boldsymbol{\lambda}^1)^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^q)]\\ &\quad+(1-\alpha)[\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}+(\boldsymbol{\lambda}^2)^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^q)]\\ &\geqslant \alpha z_{\mathrm{LR}}(\boldsymbol{\lambda}^1)+(1-\alpha)z_{\mathrm{LR}}(\boldsymbol{\lambda}^2).\end{aligned}$$

因此，$z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 为凹函数. □

定理 6.8 函数 $g(\boldsymbol{x})$ 是凹函数的充分必要条件为 $\forall\ \boldsymbol{x}^*\in\mathbf{R}^m$，存在 $\boldsymbol{s}=(s_1,s_2,\cdots,s_m)^{\mathrm{T}}\in\mathbf{R}^m$，使得

$$\forall\ \boldsymbol{x}\in\mathbf{R}^m \text{ 有 } g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x}).$$

证明 “必要性”：假设 $g(\boldsymbol{x})$ 为凹函数. 对 $H=\{(\boldsymbol{x},z)\mid z\leqslant g(\boldsymbol{x})\}$ 和 $\forall\ (\boldsymbol{x}^1,z^1),(\boldsymbol{x}^2,z^2)\in H$，有

$$\alpha(\boldsymbol{x}^1,z^1)+(1-\alpha)(\boldsymbol{x}^2,z^2)$$
$$=(\alpha\boldsymbol{x}^1+(1-\alpha)\boldsymbol{x}^2,\alpha z^1+(1-\alpha)z^2).$$

满足

$$g(\alpha\boldsymbol{x}^1+(1-\alpha)\boldsymbol{x}^2)\geqslant\alpha g(\boldsymbol{x}^1)+(1-\alpha)g(\boldsymbol{x}^2)$$
$$\geqslant\alpha z^1+(1-\alpha)z^2,$$

所以，$\alpha(\boldsymbol{x}^1,z^1)+(1-\alpha)(\boldsymbol{x}^2,z^2)\in H$，故 H 为凸集．而$(\boldsymbol{x}^*,g(\boldsymbol{x}^*))$是 H 的一个边界点，于是存在一个过$(\boldsymbol{x}^*,g(\boldsymbol{x}^*))$由法方向 $\boldsymbol{s}$ 生成的支撑超平面 $\pi:g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^*)$满足

$$g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x}).$$

必要性得证．

"充分性"：$\forall\ \boldsymbol{x}^1,\boldsymbol{x}^2,0\leqslant\alpha\leqslant1,\boldsymbol{x}^*=\alpha\boldsymbol{x}^1+(1-\alpha)\boldsymbol{x}^2$，有$g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}^1-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x}^1),g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}^2-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x}^2)$，进一步

$$g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\alpha\boldsymbol{x}^1+(1-\alpha)\boldsymbol{x}^2-\boldsymbol{x}^*)\geqslant\alpha g(\boldsymbol{x}^1)+(1-\alpha)g(\boldsymbol{x}^2),$$

即

$$g(\boldsymbol{x}^*)\geqslant\alpha g(\boldsymbol{x}^1)+(1-\alpha)g(\boldsymbol{x}^2).$$ □

图 6.4 为定理 6.7 结论的一个示意图，$z_{\mathrm{LR}}(\boldsymbol{\lambda})$是每一个极点达到最优的分段线性函数曲线

$$z_{\mathrm{LR}}(\boldsymbol{\lambda})=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^q+\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^q).$$

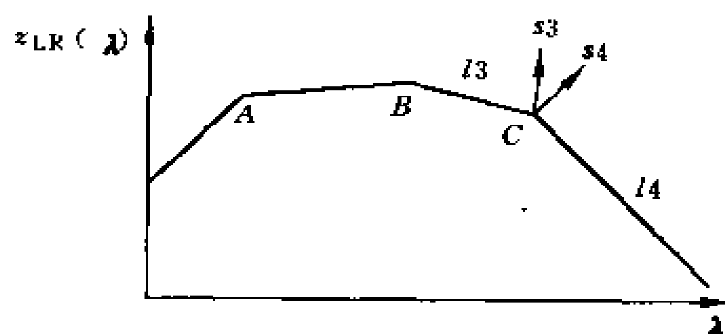


图 6.4　$z_{\mathrm{LR}}(\lambda)$函数曲线示意图

其中,每一条直线表示最优解在某一个极点达到时的目标值. 点 C 的两个方向 $s3$ 和 $s4$ 分别垂直于 $l3$ 和 $l4$. 沿 $s3$ 和 $s4$ 所夹部分的任何一个向量都可以作为满足定理 6.8 的方向而建立一个超平面. 这样的方向在 $z_{LR}(\boldsymbol{\lambda})$ 可微时是唯一的,但在诸如 A、B 和 C 点时是不唯一的. 光滑函数时的梯度法在此不再可行,因此,如下定义广义的梯度——次梯度(subgradient).

定义 6.5 若 $g:\mathbf{R}^m\to\mathbf{R}$ 为凹函数,在 $\boldsymbol{x}^*\in\mathbf{R}^m$ 向量 $\boldsymbol{s}\in\mathbf{R}^m$ 满足

$$\forall\, \boldsymbol{x}\in\mathbf{R}^m: g(\boldsymbol{x}^*)+\boldsymbol{s}^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x}),$$

则称 $\boldsymbol{s}\in\mathbf{R}^m$ 为 $g(\boldsymbol{x})$ 在 $\boldsymbol{x}^*$ 的一个次梯度. $g(\boldsymbol{x})$ 在 $\boldsymbol{x}^*$ 的所有次梯度组成的集合记为 $\partial g(\boldsymbol{x}^*)$.

定理 6.9 若 $g(\boldsymbol{x})$ 为凹函数,$\boldsymbol{x}^*$ 为 $\max\{g(\boldsymbol{x})\mid\boldsymbol{x}\in\mathbf{R}^m\}$ 最优解的充分必要条件是 $0\in\partial g(\boldsymbol{x}^*)$.

证明

$$0\in\partial g(\boldsymbol{x}^*)\Leftrightarrow 0(\boldsymbol{x}-\boldsymbol{x}^*)\geqslant g(\boldsymbol{x})-g(\boldsymbol{x}^*),\forall\,\boldsymbol{x}\in\mathbf{R}^m$$
$$\Leftrightarrow g(\boldsymbol{x})\leqslant g(\boldsymbol{x}^*),\forall\,\boldsymbol{x}\in\mathbf{R}^m.$$

□

定理 6.10 设 LR 的可行解集合 Q 由有限个整数点集合组成,它的极点为 $\boldsymbol{x}^k(k\in K)$,有

$$z_{LR}(\boldsymbol{\lambda}^*)=\min_{k\in K}\{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^k+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^k)\}.$$

记

$$I=\{i\mid z_{LR}(\boldsymbol{\lambda}^*)=\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^i+\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^i)\}.\tag{6.24}$$

则对任选 $i\in I$,

$$\boldsymbol{s}^i=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^i\tag{6.25}$$

为 $z_{LR}(\boldsymbol{\lambda})$ 在 $\boldsymbol{\lambda}^*$ 点的一个次梯度. 且

$$\Big\{\boldsymbol{s}\mid\boldsymbol{s}=\sum_{i\in I}\alpha_i\boldsymbol{s}^i,\sum_{i\in I}\alpha_i=1,\alpha_i\geqslant 0\Big\}\subseteq\partial g(\boldsymbol{\lambda}^*),$$

即所有 I 中满足(6.25)的方向凸组合形成 $\boldsymbol{\lambda}^*$ 点的次梯度子集合.

证明 若 $\boldsymbol{s}^i$ 如(6.25)得到,再由(6.24)推导出

$$(\boldsymbol{s}^i)^{\mathrm{T}}(\boldsymbol{\lambda}-\boldsymbol{\lambda}^*)=\boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^i)-\boldsymbol{\lambda}^{*\mathrm{T}}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^i)$$

$$= c^{\mathrm{T}}x^{i} + \lambda^{\mathrm{T}}(b - Ax^{i}) - [c^{\mathrm{T}}x^{i} + \lambda^{*\mathrm{T}}(b - Ax^{i})]$$
$$\geqslant z_{\mathrm{LR}}(\lambda) - z_{\mathrm{LR}}(\lambda^{*}).$$

所以,(6.25)定义的 s^{i} 为一个次梯度.

类似方法可以得到所有 I 中满足(6.25)的方向凸组合形成 λ 点的次梯度子集合. □

由定理 6.7 有关 $z_{\mathrm{LR}}(\lambda)$ 的凹性和借助图 6.4,若 λ^{*} 不是 $z_{\mathrm{LR}}(\lambda)$ 的最大值点,相交的两个达到目标值的平面 $\pi_{1}: c^{\mathrm{T}}x^{i}+\lambda^{\mathrm{T}}(b-Ax^{i})(\lambda\in D_{1})$ 和 $\pi_{2}: c^{\mathrm{T}}x^{j}+\lambda^{\mathrm{T}}(b-Ax^{j})(\lambda\in D_{2})$ 满足 $\lambda^{*}\in D_{1}\cap D_{2}$,且两个平面的法方向交角不超过 90 度,也就是 $\partial z_{\mathrm{LR}}(\lambda^{*})$ 内的所有次梯度方向夹角不超过 90 度.

定理 6.7 推出 $z_{\mathrm{LR}}(\lambda)$ 是凹函数. 由凹函数的特性,知 $z_{\mathrm{LR}}(\lambda)$ 在定义域内的任何一个开集内是连续函数. 当 λ^{*} 是光滑点时,由定理 6.7 和定理 6.10 知次梯度唯一,是函数 $z_{\mathrm{LR}}(\lambda)$ 的梯度方向,因此沿此梯度方向上升最快. 在 λ^{*} 的一个邻域内,$z_{\mathrm{LR}}(\lambda)$ 沿此方向上升.

当 λ^{*} 不是光滑点时,由 $z_{\mathrm{LR}}(\lambda)$ 的连续性和(6.24),(6.25),在 λ^{*} 的一个邻域内 $\lambda=\max\{\lambda^{*}+\theta s^{i},\mathbf{0}\}$ 时,其中 θ 是一个充分小的正数,存在 $j\in I$,使得目标函数

$$z_{\mathrm{LR}}(\lambda) = c^{\mathrm{T}}x^{j} + \lambda^{\mathrm{T}}(b - Ax^{j}),$$

于是,由 $\partial z_{\mathrm{LR}}(\lambda^{*})$ 内的所有次梯度方向夹角不超过 90 度的结论,得到

$$\begin{aligned} z_{\mathrm{LR}}(\lambda) - z_{\mathrm{LR}}(\lambda^{*}) &= (\lambda - \lambda^{*})^{\mathrm{T}}(b - Ax^{j}) \\ &\geqslant \theta(b - Ax^{i})^{\mathrm{T}}(b - Ax^{j}) \\ &\geqslant 0, \forall\, i \in I. \end{aligned}$$

上面的讨论已经提示如何构造拉格朗日松弛算法,使得 $z_{\mathrm{LR}}(\lambda)$ 逐步上升. 基本步骤是:对给定的 λ^{*} 计算 $z_{\mathrm{LR}}(\lambda^{*})$;由(6.24)和(6.25)计算 λ^{*} 的一个次梯度,若次梯度满足定理 6.9 的

充分性则达到最优解；否则以这个次梯度寻求 $z_{LR}(\boldsymbol{\lambda})$ 上升的方向.总结为算法：

次梯度优化算法

STEP1 任选一个初始拉格朗日乘子 $\lambda^1, t=1$；

STEP2 对 λ^t，从 $\partial g(\lambda^t)$ 中任选一个次梯度 s^t；若 $s^t=\mathbf{0}$，则 λ^t 达到最优解而停止计算；否则 $\lambda^{t+1}=\max\{\lambda^t+\theta_t s^t, \mathbf{0}\}$，$t := t+1$，重复 STEP2.

在 STEP2 中，θ_t 满足

$$\sum_{t=1}^{\infty}\theta_t = \infty, \text{且 } \theta_t \to 0, t \to \infty. \tag{6.26}$$

(6.26)是拉格朗日对偶问题收敛的理论结果[3].由次梯度优化算法可知，$z_{LR}(\boldsymbol{\lambda})$ 是一个凹函数，代替光滑函数的梯度上升法，它是一个次梯度上升算法.也就有非线性规划收敛的理论结果(6.26).实际计算中，不可能如同(6.26)那样迭代无穷多次，因此产生多样的确定 θ_t 的方法和具体执行次梯度优化算法的技术问题.下面就主要的问题分类讨论.

1. Q_t 的选取

实际计算的目的是尽快得到一个可以接受的下界或是对已经得到的一个解可行化，无论如何，常常采用启发式的方法.其中一类方法是

$$\theta_t = \theta_0 \rho^t, 0 < \rho < 1. \tag{6.27}$$

这类方法使 θ_t 以指数速度下降，因此迭代次数较少.另一类方法是

$$\theta_t = \frac{z_{UP}(t) - z_{LB}(t)}{\| s^t \|^2}\beta_t, \tag{6.28}$$

其中，$0 \leqslant \beta_t \leqslant 2$，一般取 $\beta_0 = 2$。当 $z_{LR}(\lambda)$上升时，β_t 不变；当$z_{LR}(\lambda)$在给定的若干步没有变化时，则取其一半.(6.28)中的$z_{UP}(t)$为 IP 最优目标值的一个上界.可以用一个可行解的目标值确定，也可以通过估计的方法得到.$z_{UP}(t)$可随 t 的变化而逐步修正.$z_{LB}(t)$是$z_{LR}(\lambda^t)$的一个下界，一般选取 $z_{LB}(t) = z_{LR}(\lambda^t)$，但有时为了计算简单，只取一个固定值.(6.28)的主要思想是用 $z_{UP}(t) - z_{LB}(t)$修正变化的速度.

2. 停止原则

(1) 迭代次数不超过 T.这是一种最为简单的原则.无论解的质量如何，到达迭代步数则停止.由此很容易控制计算的复杂性，但解的质量无法保证.

(2) $s^t = \mathbf{0} \in \partial z_{LR}(\lambda^t)$。这是最为理想的状态，由定理 6.9，$\lambda^t$ 达到拉格朗日对偶问题的最优解.在实际计算中，由于问题的复杂性和计算机本身的计算误差，这样的结果较难达到，常常用$\| s^t \| \leqslant \epsilon$($\epsilon$ 为给定的非负数)来代替.

(3) $z_{UP}(t) = z_{LB}(t)$.在 $z_{UP}(t)$和 $z_{LB}(t)$可变时，这种情况表示已得到 IP 的最优解，最优值为 $z_{IP} = z_{UP}(t) = z_{LB}(t)$.

(4) λ^t 或目标值$z_{LR}(\lambda^t)$在规定的步数内变化不超过一个给定的值.这时认为目标值不可能再变化，因此，停止运算.

具体应用中，可以采用以上停止原则之一，也可以综合运用.

6.4.2　拉格朗日启发式算法

拉格朗日松弛的一个主要工作是提供 IP 的一个下界.LR 问题的一个比较好的下界所对应的解也应该同 IP 的最优解相近.在这样的逻辑下，产生了拉格朗日启发式算法.拉格朗日松弛启发式算法主要包含两个部分.第一部分就是拉格朗日的次梯度优化计算.由于第一部分得到的 LR 的解不一定为 IP 的可行解，第二部分就是对第一部分得到的解可行化.

例 6.10 假设集合覆盖 SC 问题，通过拉格朗日松弛得到一个解 $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$，当 $\boldsymbol{x}$ 不是 SC 问题的一个可行解时，即存在 i 使得

$$\sum_{j=1}^{n} a_{ij}x_j = 0$$

时，一个直观的可行化方法是：求 k 满足

$$c_k = \min_{1\leqslant j\leqslant n}\{c_j \mid a_{ij}=1\},$$

使得 $x_k=1$.

重复以上的判别和计算直至所有的行被覆盖. 这样得到的算法是基于拉格朗日松弛的启发式算法. 所得的解为可行解. □

拉格朗日松弛启发式算法总结为：

第一阶段 次梯度优化. 这一阶段通过求解 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 满足定义 6.5 的次梯度，逐步改进 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 的值，使得 $z_{\mathrm{LR}}(\boldsymbol{\lambda})$ 与 z_{LD} 充分接近. 由启发式算法的特性，次梯度优化时不一定要求得到 z_{LD}，这样可以构造很多基于次梯度优化的启发式方法.

第二阶段 可行化. 在 LR 的解为不可行解时，对其可行化.

系数修正法就是拉格朗日松弛启发式算法的一种形式. 在给以 λ^0 后，类似次梯度优化算法，系统地调整拉格朗日系数，以改进 LR 的下界. 这一方法的优点是计算量较次梯度优化少且每一步使 LR 的下界上升. 它的缺点是所得的下界可能比较差，修正的方法依赖于问题本身. 我们先从下面的示例了解系数修正法.

例 6.11 对集合覆盖问题

$$\begin{cases} \min 2x_1 + 3x_2 + 4x_3 + 5x_4 \\ \qquad\text{s.t. } x_1 + x_3 \geqslant 1, \\ \qquad\quad x_1 + x_4 \geqslant 1, \\ \qquad\quad x_2 + x_3 + x_4 \geqslant 1, \\ x_j \in \{0,1\},\ j=1,2,3,4. \end{cases}$$

假设 $\lambda_1=1.5, \lambda_2=1.6, \lambda_3=2.2$.

$$z_{LR}(\boldsymbol{\lambda}) = \min\{-1.1x_1 + 0.8x_2 + 0.3x_3 + 1.2x_4\} + 5.3$$

$$\text{s.t.} \quad x_j \in \{0,1\},\ j = 1,2,3,4.$$

$z_{LR}(\boldsymbol{\lambda})$的最优解为 $x_1=1, x_2=x_3=x_4=0, z_{LR}(\boldsymbol{\lambda})=4.2$.

第三行没有被覆盖，在可覆盖第三行中选费用最小列

$$\delta = \min\{0.8, 0.3, 1.2\} = 0.3,$$

替代

$$\begin{bmatrix}\lambda_1\\ \lambda_2\\ \lambda_3\end{bmatrix} = \begin{bmatrix}1.5\\ 1.6\\ 2.2\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0.3\end{bmatrix} = \begin{bmatrix}1.5\\ 1.6\\ 2.5\end{bmatrix},$$

得到

$$z_{LR}(\boldsymbol{\lambda}) = \min\{-1.1x_1 + 0.5x_2 + 0.9x_4\} + 5.6 = 4.5.$$

LR 的最优解为 $x_1=x_3=1, x_2=x_4=0$. 用算法表示为

STEP0　初始化 $\boldsymbol{\lambda}^0, t=0$;

STEP2　计算 $z_{LR}(\boldsymbol{\lambda}^t)$;

STEP3　若所有行被覆盖，停止计算；否则，记 $s_i=1$ 表示第 i 行没有被覆盖，在没有被覆盖的行中任选一个行 k，计算

$$\delta_k = \min\{d_j \mid a_{kj} = 1, s_k = 1\},$$

其中 $d_j = c_j - \sum_{i=1}^{m} \lambda_i$;

STEP4　$\lambda_i^{t+1} = \begin{cases}\lambda_k^t + \delta_k, & i = k,\\ \lambda_i^t, & i \neq k,\end{cases}$ $t=t+1$，返回 STEP2.

例 6.11 不同于次梯度优化算法是迭代步长取为 1. 因此它是一种启发式的算法.　□

6.5 拉格朗日松弛在能力约束单机排序问题中的应用

柔性制造单元的排序问题是约束单机排序问题的一类，其特点是生产的多样性以满足市场的不同需求，但受生产能力的限制，必须对其生产进行调度，以期达到最优的生产效益. 加之每一个顾客对产品的需求量、完成时间等的要求不尽相同，因此我们提出一个加权目标的可拆分约束单机排序问题[4].

本节首先建立了一个加权目标约束单机排序问题的数学模型，并证明了该问题为NP-hard，然后给出一个简单的启发式算法和一个拉格朗日松弛算法并研究其最优解的性质. 最后，对这两个算法的计算结果进行了对比分析.

假设 n 种产品的外部需求为 $\{d_i,\ i=1,2,\cdots,n\}$. 加权目标约束单机排序问题的数学模型(WCS)(weighted capacitated single machine scheduling)如下：

$$Z=\min\sum_{i=1}^{n}w_iT_i \tag{6.29}$$

$$\text{s.t.}\ \sum_{t=1}^{T_i}x_{it}=d_i,\quad i=1,2,\cdots,n, \tag{6.30}$$

$$\sum_{i=1}^{n}\{a_ix_{it}+s_iY_{it}\}\leqslant c_t,$$

$$t=1,2,\cdots,\max\{T_i\,|\,i=1,2,\cdots,n\}, \tag{6.31}$$

$$Y_{it}=\begin{cases}1, & \text{若 } x_{it}>0,\\ 0, & \text{其他},\end{cases}$$

$$T_i=\max\{t\,|\,x_{it}>0\},i=1,2,\cdots,n;t=1,2,\cdots,T_i, \tag{6.32}$$

$$x_{it}\geqslant 0,i=1,2,\cdots,n,\ t=1,2,\cdots,T_i, \tag{6.33}$$

其中，w_i：产品 i 的权因子，T_i：完成需求 d_i 所需的时段，x_{it}：时段 t 产品 i 的生产批量，a_i：产品 i 的单位产品所占用的能力，s_i：产品 i 的生产准备所占用的能力，c_t：t 时段的可供能力总量.

模型 WCS 中，x_{it}，Y_{it}，T_i 为变量，x_{it}为决策变量，决定每一产品在各时段的生产批量，Y_{it}和 T_i 为 x_{it}的应变量，由(6.32)式表示.(6.29)为目标函数，使完成所有需求 $d_i(i=1, 2,\cdots, n)$所需的时段的加权平均值最小，该目标函数中的权因子是基于用户对产品需求的紧迫性、重要性等因素来决定的.(6.30)表示总产量与总需求平衡.(6.31)为能力约束方程.由(6.32)可以看出，WCS 是一个非线性的混合整数规划问题.

定理 6.11 WCS 的问题复杂性为 NP-hard.

证明 定义一个 3-partition 问题为：$3T$ 个元素满足

$$\frac{D}{4}<d_i<\frac{D}{2}, i=1,2,\cdots,3T, \sum_{i=1}^{3T}d_i=TD,$$

是否能将上面 $3T$ 个元素划分成 T 部分，使得每部分包含三个元素且三元素之和为 D?

按下列条件构造一个判定问题：对给定 T，在 WCS 中，设 $w_i=1, a_i=1, s_i=A>D>0$，产品需求为 $d_i, i=1,2,\cdots,n$. 它们满足.

$$n=3T, \sum_{i=1}^{3T}d_i=TD, \frac{D}{4}<d_i<\frac{D}{2}, c_t=D+3A, D>0.$$

是否有

$$\sum_{i=1}^{n}w_iT_i\leqslant\frac{3}{2}(1+T)T? \tag{6.34}$$

首先，非常容易验证 3-partition 问题的任何一个"是"实例是 WCS 判定问题的一个"是"实例.因为每一时段的能力约束为 $D+3A$，所以最多可以生产 3 种产品.又 $n=3T, d_i>0$，由(6.34)，必须在 T 时段内完成所有的产品加工，这样在 T 时段内每一时段正好生产 3 个产品，且不允许任一产品拆分，因为任何一个产品拆分必

增加一个生产准备占用. 于是,WCS 的一个"是"实例是 3-partition 问题的一个"是"实例. 反之,对 WCS 判定问题的任何一个"是"实例,由它的条件,限定每个时段能且只能生产三个产品,还必须在 T 个时段内完成. 由此证明 3-partition 问题多项式转换为 WCS 的判定问题. 已知 3-partition 问题是 NP-C[2],于是该 WCS 判定问题为 NP-C,因此,优化问题 WCS 为 NP-hard. □

下面构造一个简单而直观的算法. 该算法的基本思想是将 $\{w_i, i=1, 2, \cdots, n\}$ 中较大权数所对应产品尽可能早的完成,以达到目标(6.29)最小.

算法 A

STEP0 $S=\varnothing, U=\{1, 2, \cdots, n\}$. 从第一个时段 $t=1$ 开始;

STEP1 当 $U\neq\varnothing$ 时,$w_{i^*}=\max\{w_i \mid i\in U\}$,依时段 t 能力约束(6.31)情况将 i^* 尽可能往前安排直到 d_{i^*} 全部生产. 可能出现以下几种情况:

(1) 若 i^* 的全部需求 d_{i^*} 没有全部生产,且时段 t 的能力足以满足产品 i^* 的生产准备占用,则以 t 时段的最大余能力生产产品 i^*,产品 i^* 的未能加工部分到 $t+1$ 时段生产,$t=t+1$,返回 STEP1.

(2) 若 i^* 的全部需求 d_{i^*} 已全部生产,则 $S=S+\{i^*\}$. 当 $S=\{1, 2, \cdots, n\}$ 时,停止,否则 $U=U-S$,返回 STEP1.

(3) 若 i^* 的全部需求 d_{i^*} 没有全部生产,且无法在该时段生产,则 $U=U-\{i^*\}$,返回 STEP1.

STEP2 若 $U=\varnothing, S\neq\{1, 2, \cdots, n\}$,则 $t=t+1, U=\{1, 2, \cdots, n\}-S$,返回 STEP1.

这一算法实际是一种贪婪算法.从直观看,这一算法同背包问题的贪婪算法非常相像.不同的是,WCS 要求所有的外部需求都得完成,而背包问题中在能力不够时,某些物品可以不装.下例表明算法 A 的最坏界为无限.

例 6.12 设产品数 $n=6$,生产准备占用能力 $s_i=5(i=1,2,\cdots,6)$,单位产品占用能力 $a_i=1(i=1,\cdots,6)$,其他系数如下:

$$w_1=1.05,\ w_2=1.04,\ w_3=1.03,\ w_4=1.02,$$
$$w_5=1.01,\ w_6=1.00,$$
$$d_1=6, d_2=5, d_3=4, d_4=3, d_5=2, d_6=1,$$
$$c_1=25,\ c_2=26,\ c_t=5+\frac{1}{2^{t-2}}(t>2).$$

由算法 A 计算得:在 $T=1$ 时段,生产 $d_1=6, d_2=5$,能力结余 4,因为 $s_i=5(i=1,\cdots,6)$,所以无法再安排其他生产.在 $T=2$,生产 $d_3=4, d_4=3, d_5=2$,能力还余 2.同 $T=1$ 相同的道理,$d_6=1$ 只能在 $t>2$ 的时段安排.因 $s_i=5, \sum\limits_{t=3}^{\infty}\frac{1}{2^{t-2}}=1$,故有 $d_6=1$ 的完成期 $T_6\to+\infty$.从例中所给系数可以排一生产计划,在 $T=1, T=2$ 两个时段完成所有的外部需求.如 $T=1$ 时生产 $d_2=5$、$d_4=3$、$d_5=2$,$T=2$ 时生产 $d_1=6$、$d_3=4$、$d_6=1$.因此,算法 A 在最坏情况下无界. □

在给出拉格朗日松弛算法之前,先研究 WCS 的拉格朗日松弛理论.从 WCS 的模型中可以看出,能力约束(6.31)将各产品联系在一起,因此对约束(6.31)进行拉格朗日松弛.因 T_i 是一个变量,是目标函数中需要优化的变量,而拉格朗日乘子的维数为 $T=\max\{T_i|i=1,2,\cdots,n\}$,是一个变量.技术处理的方法是假设 T 充分大,即认为产品可以在有限的时段内加工完成.

对于充分大的 T,拉格朗日松弛后的数学模型 LRP 是:

$$\text{LRP}\begin{cases} Z & = \max\limits_{\lambda} Z(\boldsymbol{\lambda}) \\ & = \max\limits_{\lambda}\min\limits_{X}\left\{\sum\limits_{i=1}^{n} w_i T_i + \sum\limits_{t=1}^{T}\lambda_t\left\{\sum\limits_{i=1}^{n}[a_i x_{it} + s_i Y_{it}] - c_t\right\}\right\} \\ \text{s.t.} & (6.30),(6.32),(6.33) 和 \lambda_t \geqslant 0. \end{cases}$$

其中，$\boldsymbol{X}=(x_{it})_{n\times T}$，一个 $n\times T$ 的变量矩阵，$\boldsymbol{\lambda}=(\lambda_1,\lambda_2,\cdots,\lambda_T)$.

由于松弛了(6.31)，(6.30)、(6.32)和(6.33)所决定的模型 LRP 是无能力约束的单机排序问题，对给定充分大的 T，各产品的完工时间都可在 T 时段内完成，即 $1\leqslant T_i\leqslant T$. 因拉格朗日松弛后目标函数的惩罚性，使得 $T(1\leqslant T_i\leqslant T)$在一个充分大的范围内取值. 在技术处理 LRP 时，T 可以从 1 逐步增加. 对给定的 T，等价于当 $t>T$ 取 $\lambda_t=0$.

记

$$g(\boldsymbol{X},T,\boldsymbol{\lambda}) = \sum_{i=1}^{n} w_i T_i + \sum_{t=1}^{T}\lambda_t\left\{\sum_{i=1}^{n}[a_i x_{it} + s_i Y_{it}] - c_t\right\}, \tag{6.35}$$

则(6.35)可以改写为下列形式：

$$g(\boldsymbol{X},T,\boldsymbol{\lambda}) = \sum_{i=1}^{n}\left[w_i T_i + \sum_{t=1}^{T}\lambda_t a_i x_{it} + \sum_{t=1}^{T}\lambda_t s_i Y_{it}\right] - \sum_{t=1}^{T}\lambda_t c_t, \tag{6.36}$$

再记

$$h(i,\boldsymbol{X},T,\boldsymbol{\lambda}) = w_i T_i + \sum_{t=1}^{T}\lambda_t a_i x_{it} + \sum_{t=1}^{T}\lambda_t s_i Y_{it}, \tag{6.37}$$

由(6.35)，(6.36)和(6.37)可推出

$$\begin{aligned} \max_{\lambda} Z(\boldsymbol{\lambda}) &= \max_{\lambda}\min_{X}\left\{\sum_{i=1}^{n} w_i T_i + \sum_{t=1}^{T}\lambda_t\left\{\sum_{i=1}^{n}[a_i x_{it} + s_i Y_{it}] - c_t\right\}\right\} \\ &= \max_{\lambda}\min_{X} g(\boldsymbol{X},T,\boldsymbol{\lambda}) \\ &= \max_{\lambda}\left\{\sum_{i=1}^{n}\min_{X} h(i,\boldsymbol{X},T,\boldsymbol{\lambda}) - \sum_{t=1}^{T}\lambda_t c_t\right\}. \end{aligned}$$

LRP 可以表示如下：

$$\begin{cases} Z = \max\limits_{\lambda}\left\{\sum\limits_{i=1}^{n}\min\limits_{X} h(i,\boldsymbol{X},T,\boldsymbol{\lambda}) - \sum\limits_{t=1}^{T}\lambda_t c_t\right\} \\ \text{s.t.}\quad (6.30),(6.32),(6.33) \text{ 和 } \boldsymbol{\lambda}\geqslant \boldsymbol{0}. \end{cases} \tag{6.38}$$

求解 LRP 就可分成两步. 首先，当 $\boldsymbol{\lambda}\geqslant\boldsymbol{0}$ 为定数时求解下列子问题：

SUBP

$$\begin{cases} \min\limits_{X} h(i,\boldsymbol{X},T,\boldsymbol{\lambda}) = \min\limits_{Y}\left\{w_i T_i + \sum\limits_{t=1}^{T}\lambda_t a_i x_{it} + \sum\limits_{t=1}^{T}\lambda_t s_i Y_{it}\right\} \\ \qquad \text{s.t.}\quad (6.30),(6.32) \text{ 和} (6.33), \end{cases} \tag{6.39}$$

得

$$Z(\boldsymbol{\lambda}) = \sum_{i=1}^{n}\min_{X} h(i,\boldsymbol{X},T,\boldsymbol{\lambda}) - \sum_{t=1}^{T}\lambda_t c_t, \tag{6.40}$$

而后，再从所有的 $\boldsymbol{\lambda}\geqslant\boldsymbol{0}$ 中对(6.40)求最大值.

定理 6.12　对给定充分大的 T，若 $\lambda_t(t=1,2,\cdots,T)$ 已知且非负，则 SUBP 一定有下列形式的最优解，

$$x_{it} = \begin{cases} d_i, & \text{存在某一 } t^*, 0\leqslant t = t^* \leqslant T, \\ 0, & t\neq t^*. \end{cases}$$

证明　设 $x_{it}^*(t=1,2,\cdots,T)$ 为 SUBP 的最优解. 设 $t=t_1,t_2,\cdots,t_m$ 时，$x_{it}^*>0$. 令 $\lambda_{t^*}=\min\{\lambda_t \mid x_{it}>0\}$. 根据(6.39)，很易得

$$\begin{aligned} &\sum_{t=1}^{T}\lambda_t a_i x_{it} + \sum_{t=1}^{T}\lambda_t s_i Y_{it} \\ &\geqslant \lambda_{t^*} a_i d_i + \lambda_{t^*} s_i Y_{it^*} \\ &\geqslant \lambda_{t^*} a_i d_i + \lambda_{t^*} s_i, \end{aligned}$$

且由 $w_i T_i \geqslant w_i t^*$，推出，

$$x_{it}^{**} = \begin{cases} d_i, & t = t^*, \\ 0, & t\neq t^*, \end{cases}$$

为最优解. 结论得证. 文中后续的拉格朗日算法正是基于此结论，

逐步迭代逼近最优解. □

由定理 6.12,求解 SUBP 的最优解只需判断

$$w_i t^* + \lambda_{t^*}\{a_i d_i + s_i\} \leqslant w_i t + \lambda_t\{a_i d_i + s_i\}, 0 \leqslant \forall\, t \leqslant T. \tag{6.41}$$

计算出(6.41)的 t^* 最多需要 T^2 次比较,所以当 $\boldsymbol{\lambda}$ 确定时,求 SUBP 的最优解相当容易. 此时,可得到 $\{x_{it}, Y_{it}, i=1, 2, \cdots, n, t=1, 2, \cdots, T\}$. 我们还必须求解 $\boldsymbol{\lambda}$ 使得 $Z(\boldsymbol{\lambda})$ 接近或等于 Z. 当 $\boldsymbol{X}=\{x_{it} \mid i=1,2,\cdots,n; t=1,2,\cdots,T\}$ 为满足(6.41)的最优解时,下列向量 $\boldsymbol{\nu}(\boldsymbol{X})=(v(1,\boldsymbol{X}), v(2,\boldsymbol{X}), \cdots, v(T, \boldsymbol{X}))$

$$\boldsymbol{\nu}(t,\boldsymbol{X}) = \left\{\sum_{i=1}^{n}[a_i x_{it} + s_i Y_{it}] - c_t\right\}^+, \quad t = 1,2,\cdots,T, \tag{6.42}$$

为 LRP 满足(6.25)的次梯度(练习题),其中,$x^+=\begin{cases} x, & x \geqslant 0, \\ 0, & \text{其他}. \end{cases}$

采用(6.28)的迭代步长法和以下两条判停准则:

(a) 给定迭代步数上限;

(b) 给定充分小的 $\varepsilon>0$, $\sum_{t=1}^{T} v^2(t, \boldsymbol{X}^k) < \varepsilon$ 时.

由拉格朗日松弛判停后,所得到的解 $\boldsymbol{X}$ 不一定是 WCS 的可行解. 当为 WCS 的不可行解时,需进行解的可行性修改. 修改的原则是,若时段 t 的排序超过其能力,尽量将权数大的所对应的产品前移到尚有余能力的时段生产. 若前移不行,则尽可能将权数小的产品后移至有能力余量的一个时段生产,以达解的可行. 若前两方案都达不到可行解,则以再增加时段来完成.

算法 B

STEP0 $T=1, \lambda^0(1)=0, Z^*$ 为算法 A 求得的目标值, $k=0$.

STEP1 解 SUBP,以(6.41)分别求出 SUBP 中每一产品的最优解,再以(6.42)求次梯度 $\boldsymbol{\nu}(\boldsymbol{X})$,最后由(6.28)求解 θ_k,

$\lambda^{k+1}=\max\{0,\lambda^k+\theta_k y(X)\}$. 若不满足判停准则，

(1) 当 $\lambda^{k+1}(T)=0$ 时，$k=k+1$，返回 STEP1；

(2) 当 $\lambda^{k+1}(T)>0$ 时，$T=T+1$，$\lambda^{k+1}(T)=0$，$k=k+1$，返回 STEP1；若满足判停准则，到 STEP2.

STEP2　若所求解为 WCS 的可行解，停止. 否则，按上面讨论进行可行性处理.

以上算法在 PC-386 采用 Turbo C 编程，分别就 $n=30$ 个产品，将产品需求分为四类，每类各产生 10 个正态分布的数据文件，其正态分布的均值和方差见表 6.1. 每一时段的能力约束分别为 w_i 与 s_i 均值之和的 1,2,4,6 倍，a_i 都取 1. 因此，由四类产品需求及四类能力约束的组合，共 16 组. 每组 10 个数据文件，共 160 个数据文件进行验证.

表 6.1　数据分类

项目	均值	方差	项目	均值	方差
权因子 w	50	10	产品需求 2	10	2
生产准备 s	10	2	产品需求 3	40	10
产品需求 1	5	1	产品需求 4	100	10

计算中，拉格朗日松弛的判停迭代次数为 200 次，$\epsilon=0.01$. 计算结果见表 6.2，其中，比例$(a:b)$为能力约束与 s_i 和 d_i 两项均值和的比. Z_A 为算法 A 的计算结果，Z_{LB}为算法 B 的拉格朗日松弛下界，Z_B为算法 B 可行化后的最终结果. 表中 Z_A，Z_B，Z_{LB}为对应 10 个数据文件计算结果的平均值，$GAP1$ 和 $GAP2$ 按如下定义

$$GAP1=\frac{Z_B-Z_{LB}}{Z_{LB}},\quad GAP2=\frac{Z_A-Z_B}{Z_A-Z_{LB}}.$$

从计算结果可以看出，当每一时段能力平均意义下只能完成一个或两个产品时(1∶1 或 2∶1 部分)，算法 A 的结果比较好，八

组中，有四组结果好于算法 B，一组相同，三组次于算法 B，从直观解释，此时可根据权因子大小排序. 但当每时段可完成多类产品时，算法 B 的结果好于算法 A，八组都不次于算法 A，特别是第十二组差值高达 0.53. 同时还可以看出，当每时段可完成产品类越多时，拉格朗日松弛算法所得下界 Z_{LB}和 Z_B 的相对误差变得越小. 从计算结果综合来说，拉格朗日松弛算法对能力约束单机排序问题是比较适用的.

表 6.2 计算结果的平均值

组数(比例)	Z_A	Z_{LB}	Z_B	$GAP1$	$GAP2$
1(1:1)	35047.40	20322.76	31008.30	0.53	0.27
2 (1:1)	32969.50	20167.81	34853.10	0.73	−0.15
3 (1:1)	25631.60	19267.57	32511.30	0.69	−1.08
4 (1:1)	25018.20	20249.87	28256.20	0.40	−1.89
5 (2:1)	13987.60	10440.72	13747.10	0.32	0.07
6 (2:1)	13073.20	10430.44	13644.60	0.31	−0.22
7 (2:1)	11943.60	10195.26	11924.10	0.17	0.01
8 (2:1)	11383.80	10527.58	11382.20	0.08	0.00
9 (4:1)	6504.60	5646.47	6246.80	0.11	0.30
10 (4:1)	6361.10	5655.45	6247.40	0.10	0.16
11 (4:1)	6100.40	5518.72	5915.40	0.07	0.32
12 (4:1)	6016.90	5675.72	5837.60	0.03	0.53
13 (6:1)	4406.00	4045.23	4309.20	0.07	0.27
14 (6:1)	4376.70	4047.10	4380.80	0.08	0
15 (6:1)	4303.90	3957.02	4188.80	0.06	0.33
16 (6:1)	4255.90	4054.58	4224.00	0.04	0.16

练 习 题

1. 证明：(6.42)是 LRP 满足(6.25)的次梯度.

2. 与松弛相对应的一个概念是限制. 整数规划 IP 的一个限制问题

$$Z_R = \min\{Z_R(\boldsymbol{x}) \mid \boldsymbol{x} \in S_r\}$$

满足(a)$S_R \subseteq S = \{\boldsymbol{x} \in \boldsymbol{Z}_+^n \mid \boldsymbol{A}\boldsymbol{x} \geqslant \boldsymbol{b}, \boldsymbol{B}\boldsymbol{x} \geqslant \boldsymbol{d}\}$,(b)$Z_R(\boldsymbol{x}) \geqslant \boldsymbol{c}^T\boldsymbol{x}$.

(1) 通过什么样的处理方法可以实现 IP 的限制?

(2) 能得到怎样的结论?

3. 整数规划问题

$$\min 7x_1 + 6x_2 + 2x_3$$

$$\text{s.t.} \begin{bmatrix} 3 \\ 3 \end{bmatrix} x_1 + \begin{bmatrix} 3 \\ 1 \end{bmatrix} x_2 + \begin{bmatrix} 1 \\ 2 \end{bmatrix} x_3 \geqslant b,$$

$$\boldsymbol{x} \in \boldsymbol{Z}_+^3.$$

分别用图解的方法就以下条件求上述整数规划问题的拉格朗日对偶最优值.

(1) $\boldsymbol{b} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$时,拉格朗日松弛第一个约束;

(2) $\boldsymbol{b} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$时,拉格朗日松弛第二个约束的值;

(3) (1)和(2)的值是否有差别? $\boldsymbol{b}$ 为多少时,(1)和(2)的值相等且与上面整数规划的目标值相等?

4. 整数规划问题

$$\max 2x_1 + 5x_2$$

$$\text{s.t.} \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ 4 \\ -1 \end{bmatrix} x_2 \leqslant \begin{bmatrix} 28 \\ 27 \\ 1 \end{bmatrix}, \boldsymbol{x} \in \boldsymbol{Z}_+^2.$$

对上面整数规划问题按以下各条件分别讨论:

(1) 证明:采用拉格朗日松弛上述整数规划中的任何两个约束,则拉格朗日对偶值等于上述整数规划的线性规划松弛的最优值.

(2) 寻找一个目标函数使得(1)不成立.

(3) 证明:拉格朗日松弛任何一个约束的拉格朗日对偶值都是线性规划松弛的改进.

(4) 将(1)~(3)用图解法表示出来.

5. 就下面整数规划问题分别松弛两个约束的两种拉格朗日松弛方法

$$\max \sum_i \sum_j c_{ij} x_{ij}$$
$$\text{s.t.} \quad \sum_j x_{ij} \leqslant 1, i \in M,$$
$$\sum_i l_i x_{ij} \leqslant b_j, j \in N,$$
$$\boldsymbol{x} \in \{0,1\}^{|M| \times |N|}.$$

比较它们在以下各方面的优劣势:

(1) 松弛后的子问题是否容易求解?

(2) 拉格朗日对偶是否容易求解?

(3) 拉格朗日对偶值的优劣?

6. 就下面整数规划问题分别松弛两个约束的两种拉格朗日松弛方法

$$\min \sum_{i \in M} \sum_{j \in N} h_{ij} y_{ij} + \sum_{j \in N} c_j x_j$$
$$\text{s.t.} \quad \sum_j y_{ij} \leqslant a_i, \quad i \in M,$$
$$\sum_i y_{ij} \leqslant b_j x_j, j \in N,$$
$$y_{ij} \leqslant \min(a_i, b_j) x_j, i \in M, j \in N$$
$$\boldsymbol{y} \in \mathbf{R}_+^{mn}, \boldsymbol{x} \in \{0,1\}^{|N|}.$$

讨论拉格朗日松弛各种约束的优劣性.

7. 考虑无约束单机排序问题:n 个工件在一台机器上加工,p_j、r_j 和 w_j 分别是工件 j 的加工时间、最早开工时间和权数. 这些是已知参数. 记 t_j 为工件 j 的开工时间. 极小化加权开工时间的单

机排序问题，使 $\sum_{1\leq j\leq n} w_j t_j$ 最小. 若没有最早开工时间的限制，即所有 $r_j=0$，最优加工顺序是按 $\frac{w_j}{p_j}(j=1,2,\cdots,n)$ 从大到小加工. 如何建立这个问题的数学模型和进行拉格朗日松弛以得到问题的一个下界？

8. 考虑第四章 4.6 节的生产批量问题，怎样应用拉格朗日松弛得到问题的一个下界？怎样构造一个基于拉格朗日松弛的启发式算法，以得到问题的一个可行解？

9. 通过数值计算的方法验证对练习 7 的分析结果.

10. 就关心的组合优化问题应用拉格朗日松弛算法.

参考文献

1. Khachian L G. A polynomial algorithm for linear programming. Doklady Akad. Nauk USSR, 1979, 244(5):1093～1096
2. Garey M R, Johnson D S. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979
3. Fisher M L. The Lagrangian relaxation method for solving integer programming problems, Management Science, 1981, 27(1):1～18
4. Xing W, Zhang J, Jiang Q et al. Capacitated single flexible manufacturing cell with setups: model, complexity and Lagrangean relaxation. In: Ding-Zhu Du et al ed. Operations Research and Its Applications. 1995, 162～170

索引及英文关键词

(1) 按笔划顺序排列.
(2) m.n 指第 m 单第 n 节.

十六划